Business Case: Walmart - Confidence Interval and CLT

```
Walmart is an American multinational retail corporation that operates a chain of supercenters,
discount departmental stores, and grocery stores from the United States. Walmart has more than
100 million customers worldwide.
```

Business Problem

```
The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically,
purchase amount) against the customer’s gender and the various other factors to help the business
make better decisions. They want to understand if the spending habits differ between male and female
customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and
50 million are female).
```

Problem Statement and Analyzing basic metrics

```
    #Problem Statement:
    Walmart has begin journey as as small discount retailer in Rogers,Ark. It has opened thousands of
    stores in the U.S. and expanded internationally.
    Through technology, they created seamless experience to let customers shop anytime and anywhere online
    and in stores. But as expansion increases, it is very important to have growth in customer bases also.
    In order to compete with other multinational brands, it needs to ramp up customer base by studying
    customers needs in detail.
    Company needs to analyse customer base so that their products can be sell at faster rate with maximum
    quantity.

#Basic metrics:
1. Know Your Customer:
   Customers are the pillars of any business. It is very important to know about your customer,
   their likes and dislikes. It is very important to know whether males are purchasing more or females,
   whether married couples are making large customer base or single customers, or which age group is
   highly active in buying products.
```

```
2. Increase Customer base:
   Customer base is one of the most important metric. It directly impacts the business revenue and
   the position hold in the market. Walmart needs to attract more customers. As customer base
   increases, data will be more and hence more information leads to better insights and therefore
   can help in taking better decisions.

3. Customer retention:
   While attracting new customers, it is also important to retain the existing customers. The data
   for existing customers leads to some insights which in turn can increase addition of new customers.

4. Popularity of product category:
   It is important to know the popularity of product category for different customer base.

5. Customer Enagagement in Product:
   It is important to know how much customer is giving time to company's product. If the customer made
   purchase of any product once or twice, there will be no increase in demand of that product. It is
   necessary to know, how much customer is engaging with the company's product.

6. Revenue and Marketing:
   It is important to understand how much revenue has been generated from various category products.
   Metrics such as average revenue per customer and lifetime customer value are too important for any
   business. They can drive the business into new directions and can help in taking better decisions
   for pricing and marketing strategy to get maximum revenue.
```

In [1]:

```python
#importing all libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy import stats

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#read the Walmart dataset csv file
data = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?
```

In [3]:
```
#showing first five records of dataset
data.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

In [4]:
```
#Observations on shape of data
data.shape
print(f"Number of rows: {data.shape[0]}\nNumber of columns: {data.shape[1]}")
```

```
Number of rows: 550068
Number of columns: 10
```

```
1. The dataset contains 550068 records of customers along with 10 attributes.
2. In simpler terms, there are 550068 rows and 10 columns.
```

In [5]:
```
#basic information about dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

```
1. The columns in dataset are "User_ID","Product_ID","Gender","Age","Occupation","City_Category",
   "Stay_In_Current_City_Years","Marital_Status","Product_Category" and "Purchase".
2. Fields such as "Product_ID","Gender","Age","City_Category" and "Stay_In_Current_City_Years" are
   of Object datatype and contains categorical data.
   While fields like "User_ID", "Occupation","Marital_Status","Product_Category" and "Purchase" are
   int64 datatype.
3. There are 550068 records for each field.
4. "Marital_Status" is a field which is of integer type. However, it should be an object datatype.
```

Conversion of categorical attribute "Marital_Status" int datatype to object datatype

In [6]:
```
# Values of those fields
print("min and max value of field 'Marital Status' is :",data["Marital_Status"].min(),
      "and",data["Marital_Status"].max(),'respectively')
print("min and max value of field 'Occupation' is :",data["Occupation"].min(),
      "and",data["Occupation"].max(),'respectively')
print("min and max value of field 'Product_Category' is :",data["Product_Category"].min(),
      "and",data["Product_Category"].max(),'respectively')

# It can be seen that Marital_Status has been categorized in integer datatype. However, it can be
# categorized as an object type. So, Marital status as 1 represents that customer is married and
# with 0, they are considered as single.
# As field "Occupation" and "Product_Category" are masked. It is quite observable that these fields
# contain categorical values. Therefore, analysis of parameters like mean, variance etc for this field
# makes no sense.
```

```
min and max value of field 'Marital Status' is : 0 and 1 respectively
min and max value of field 'Occupation' is : 0 and 20 respectively
min and max value of field 'Product_Category' is : 1 and 20 respectively
```

In [7]:
```
data['Marital_Status'] = data['Marital_Status'].replace([1, 0], ['married', 'single'])
data['Occupation'] = data['Occupation'].astype(str)
data['Product_Category'] = data['Product_Category'].astype(str)
data.head(2)
```

Out[7]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	single	3	8370
1	1000001	P00248942	F	0-17	10	A	2	single	1	15200

In [8]:
```
# Check again data types of all the attributes
data.dtypes
```

Out[8]:
```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                    object
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                object
Product_Category              object
Purchase                       int64
dtype: object
```

In [9]:
```
#statistical summary of columns containing categorical data
data.describe(include = object).T

# There are 3631 unique products in dataset.
```

Out[9]:

	count	unique	top	freq
Product_ID	550068	3631	P00265242	1880
Gender	550068	2	M	414259
Age	550068	7	26-35	219587
Occupation	550068	21	4	72308
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821
Marital_Status	550068	2	single	324731
Product_Category	550068	20	5	150933

In [10]:
```
#statistical summary of columns containing numerical data
data.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	1004478.0	1006040.0
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	12054.0	23961.0

Detection of missing values

In [11]:
```
data.isna().sum()

#It can be observed that none of the fields of dataset have missing values.
```

Out[11]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

In [12]:
```
from IPython.display import display, HTML
CSS = """
.output {
    flex-direction: row;
}
"""
HTML('<style>{}</style>'.format(CSS))
```

Out[12]:

In [13]:
```
#Analysis data of Gender column
d1 = pd.DataFrame(data["Gender"].value_counts().reset_index())
d1.columns = ["Gender","Count"]
d1["% count"] = round((d1["Count"]/len(data))*100,2)

#Analysis data of Marital_Status column
d2 = pd.DataFrame(data["Marital_Status"].value_counts().reset_index())
d2.columns = ["Marital_Status","Count"]
d2["% count"] = round((d2["Count"]/len(data))*100,2)

display(d1)
display(d2)
```

	Gender	Count	% count
0	M	414259	75.31
1	F	135809	24.69

	Marital_Status	Count	% count
0	single	324731	59.03
1	married	225337	40.97

```
1. There are 75.31% male customers while only 24.69% female customers
2. There are 59.03% single customers while only 40.97% married customers
```

In [14]:
```
#Analysis data of City_Category column
d3 = pd.DataFrame(data["City_Category"].value_counts().reset_index())
d3.columns = ["City_Category","Count"]
d3["% count"] = round((d3["Count"]/len(data))*100,2)

#Analysis data of Stay_In_Current_City_Years column
d4 = pd.DataFrame(data["Stay_In_Current_City_Years"].value_counts().reset_index())
d4.columns = ["Stay_In_Current_City_Years","Count"]
d4["% count"] = round((d4["Count"]/len(data))*100,2)

display(d3)
display(d4)
```

	City_Category	Count	% count
0	B	231173	42.03
1	C	171175	31.12
2	A	147720	26.85

	Stay_In_Current_City_Years	Count	% count
0	1	193821	35.24
1	2	101838	18.51
2	3	95285	17.32
3	4+	84726	15.40
4	0	74398	13.53

```
In [15]: #Analysis data of Occupation column
         d5 = pd.DataFrame(data["Occupation"].value_counts().reset_index())
         d5.columns = ["Occupation","Count"]
         d5["% count"] = round((d5["Count"]/len(data))*100,2)

         #Analysis data of Product_Category column
         d6 = pd.DataFrame(data["Product_Category"].value_counts().reset_index())
         d6.columns = ["Product_Category","Count"]
         d6["% count"] = round((d6["Count"]/len(data))*100,2)

         display(d5)
         display(d6)
```

	Occupation	Count	% count
0	4	72308	13.15
1	0	69638	12.66
2	7	59133	10.75
3	1	47426	8.62
4	17	40043	7.28
5	20	33562	6.10
6	12	31179	5.67
7	14	27309	4.96
8	2	26588	4.83
9	16	25371	4.61
10	6	20355	3.70
11	3	17650	3.21
12	10	12930	2.35
13	5	12177	2.21
14	15	12165	2.21
15	11	11586	2.11
16	19	8461	1.54
17	13	7728	1.40
18	18	6622	1.20
19	9	6291	1.14
20	8	1546	0.28

	Product_Category	Count	% count
0	5	150933	27.44
1	1	140378	25.52
2	8	113925	20.71
3	11	24287	4.42
4	2	23864	4.34
5	6	20466	3.72
6	3	20213	3.67
7	4	11753	2.14
8	16	9828	1.79
9	15	6290	1.14
10	13	5549	1.01
11	10	5125	0.93
12	12	3947	0.72
13	7	3721	0.68
14	18	3125	0.57
15	20	2550	0.46
16	19	1603	0.29
17	14	1523	0.28
18	17	578	0.11
19	9	410	0.07

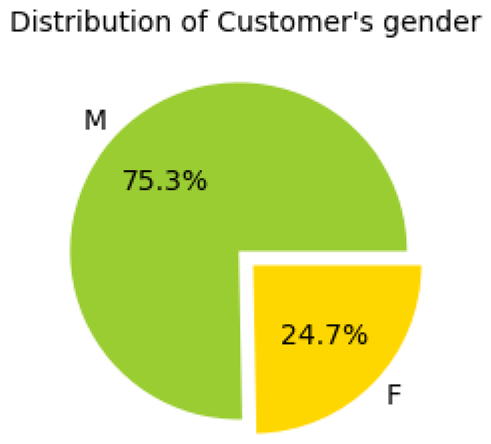
In [16]:
```
#Analysis data of Age column
d7 = pd.DataFrame(data["Age"].value_counts().reset_index())
d7.columns = ["Age","Count"]
d7["% count"] = round((d7["Count"]/len(data))*100,2)
d_age = d7.sort_values(by = "Age")
display(d_age)
```

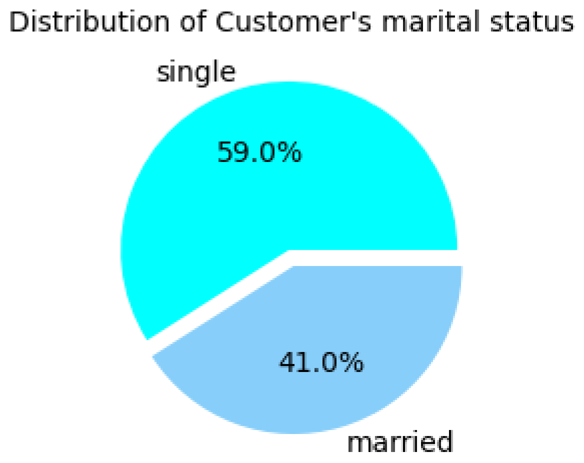
	Age	Count	% count
6	0-17	15102	2.75
2	18-25	99660	18.12
0	26-35	219587	39.92
1	36-45	110013	20.00
3	46-50	45701	8.31
4	51-55	38501	7.00
5	55+	21504	3.91

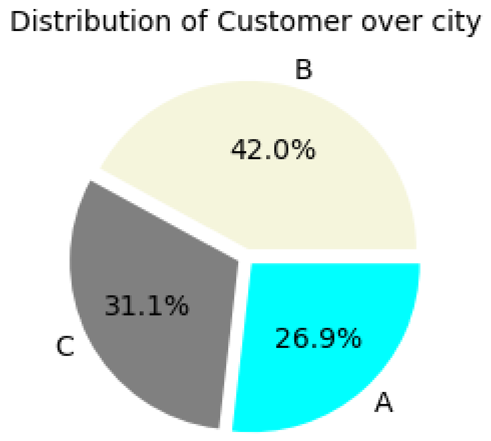
```
1. Maximum % of customers belong to age group 26-35.
2. Minimum % of customers belong to age group 0-17.
```

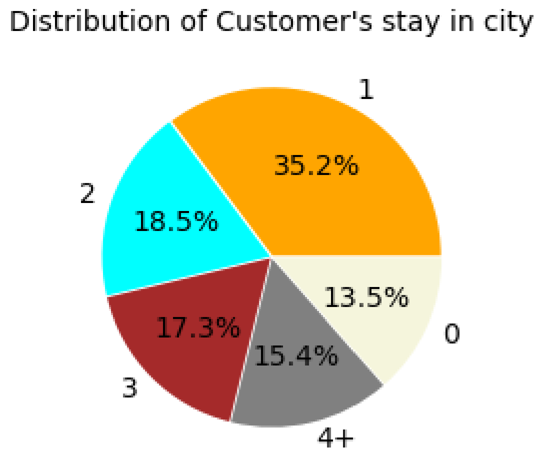
Percentage distribution of customer over various fields

In [17]:
```python
plt.figure(figsize = (9,6))
explode_gender = (0.06,0.06)
explode_mar = (0.05,0.05)
explode_city = (0.05,0.05,0.05)
explode_stay = (0.01,0.01,0.01,0.01,0.01)
colors_gender = ['yellowgreen','gold']
colors_mar = ["cyan", 'lightskyblue']
colors_city = ['beige',"grey", "cyan"]
colors_stay = ["orange", "cyan", "brown","grey", "beige"]

plt.subplot(2,2,1)
plt.pie(d1["Count"], labels=d1["Gender"], explode=explode_gender,colors = colors_gender, autopct='%1.1f%%')
plt.title("Distribution of Customer's gender", fontsize = 10)

plt.subplot(2,2,2)
plt.pie(d2["Count"], labels=d2["Marital_Status"], explode=explode_mar,colors = colors_mar,
        autopct='%1.1f%%')
plt.title("Distribution of Customer's marital status", fontsize = 10)

plt.subplot(2,2,3)
plt.pie(d3["Count"], labels=d3["City_Category"], explode=explode_city,colors = colors_city,
        autopct='%1.1f%%')
plt.title("Distribution of Customer over city", fontsize = 10)

plt.subplot(2,2,4)
plt.pie(d4["Count"], labels=d4["Stay_In_Current_City_Years"], explode=explode_stay,colors = colors_stay,
        autopct='%1.1f%%')
plt.title("Distribution of Customer's stay in city", fontsize = 10)

plt.show()
```

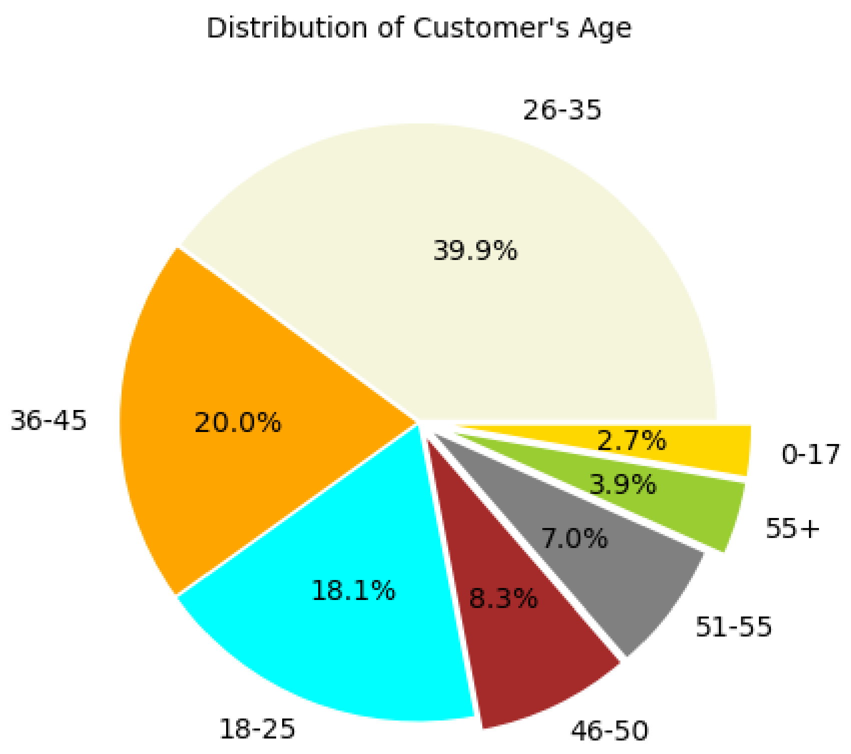

Distribution of Customer's gender
M
75.3%
24.7%
F


Distribution of Customer's marital status
single
59.0%
41.0%
married


Distribution of Customer over city
B
42.0%
31.1%
C
26.9%
A


Distribution of Customer's stay in city
1
35.2%
2
18.5%
13.5%
0
17.3%
15.4%
3
4+

In [18]:
```
explode_age = (0.01,0.01,0.01,0.06,0.06,0.12,0.12)
colors_age = ["beige","orange", "cyan", "brown","grey",'yellowgreen','gold']
plt.figure()
plt.pie(d7["Count"], labels=d7["Age"], explode=explode_age,colors = colors_age,
        autopct='%1.1f%%')
plt.title("Distribution of Customer's Age", fontsize = 10)
plt.show()
```

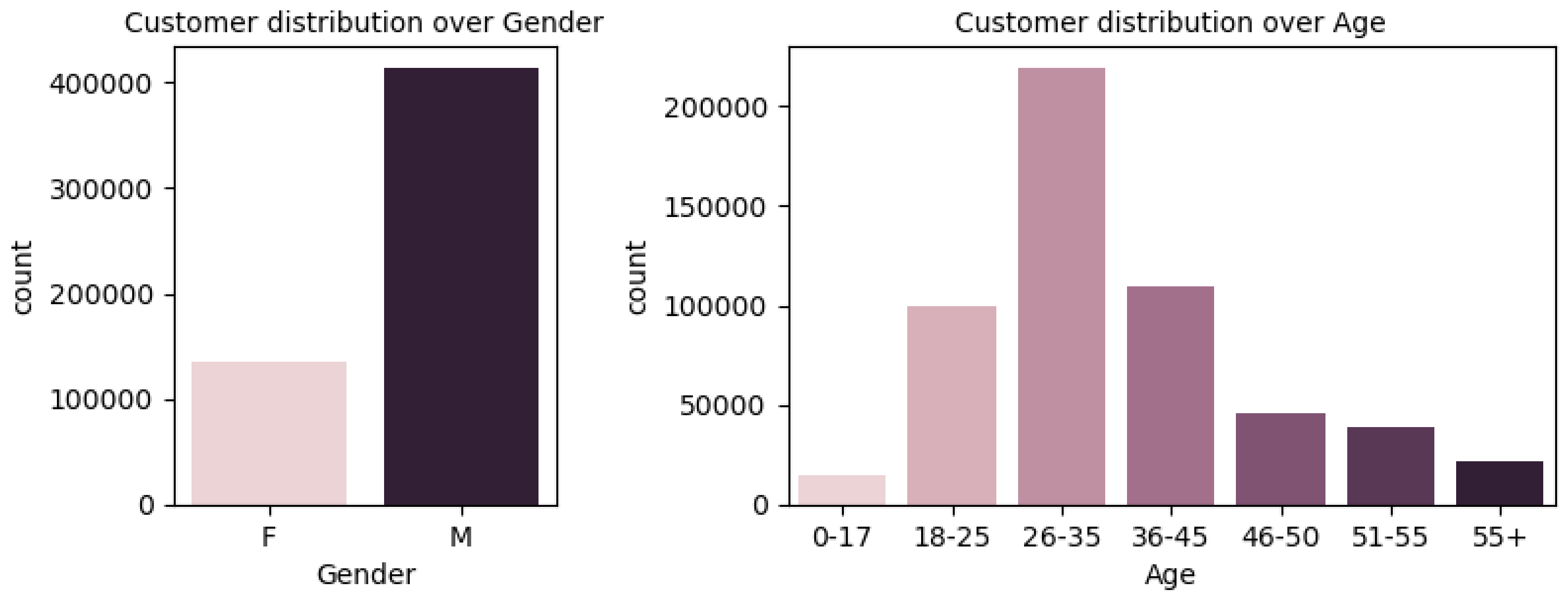


From the above plot, it can be seen that maximum customers are from age group 26-35, followed by age-group 36-45 and then 18-25.
This can be inferred that young and middle-aged adults are more likely to expense in comparison with others.

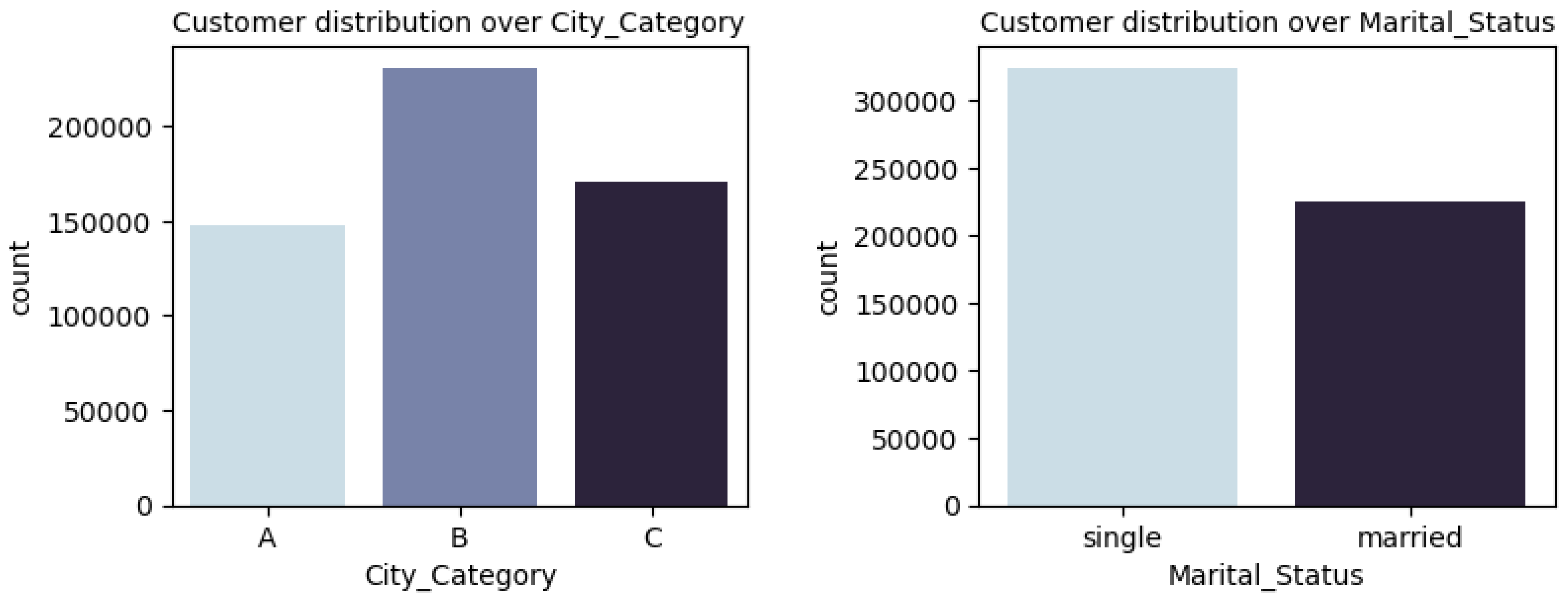
```
The age-group 46-50 and 51-55 are somewhat around 7%-8%.
```

Absolute distribution of customers over different fields or columns

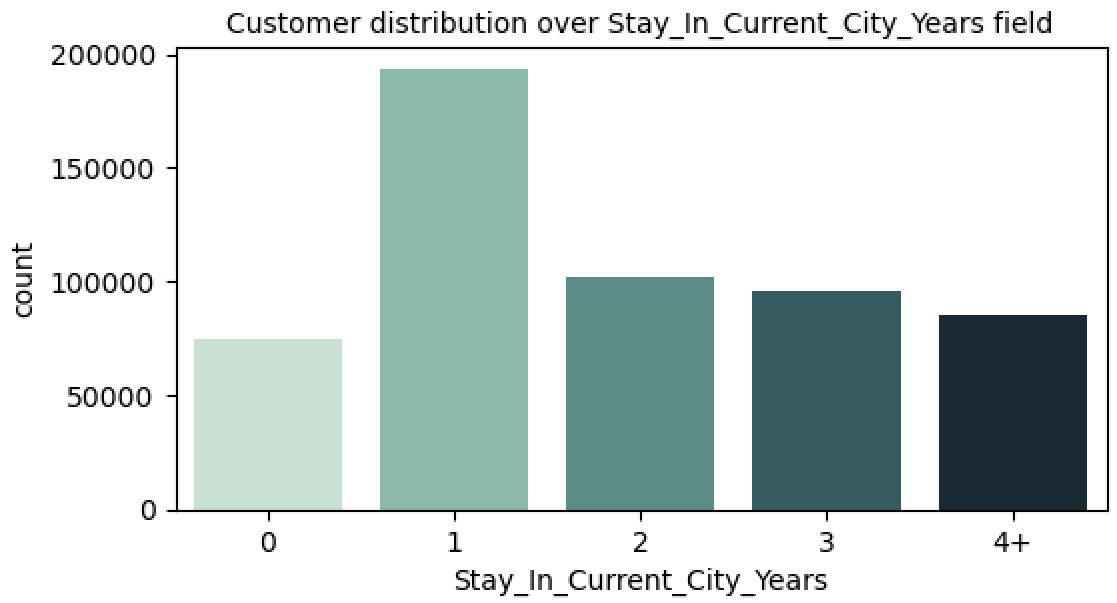
In [19]:
```
#Analysis data of Gender and Age column
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3),gridspec_kw={'width_ratios': [1, 2]})
fig.subplots_adjust(wspace = 0.4)
sns.countplot(data = data, x = data["Gender"],palette='ch:s=.25,rot=.25',ax=axis[0])
axis[0].set_title("Customer distribution over Gender", pad=5, fontsize=10)
sns.countplot(data = data, x = data["Age"].sort_values(),palette='ch:s=.25,rot=.25',ax=axis[1])
axis[1].set_title("Customer distribution over Age", pad=5, fontsize=10)
plt.show()
```



In [20]:
```
#Analysis data of City_Category and Marital_Status column
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3),gridspec_kw={'width_ratios': [1, 1]})
fig.subplots_adjust(wspace = 0.4)
sns.countplot(data = data, x = data["City_Category"].sort_values(),palette='ch:s=.25,rot=-.25',ax=axis[0])
axis[0].set_title("Customer distribution over City_Category", pad=5, fontsize=10)
sns.countplot(data = data, x = data["Marital_Status"],palette='ch:s=.25,rot=-.25',ax=axis[1])
axis[1].set_title("Customer distribution over Marital_Status", pad=5, fontsize=10)
plt.show()
```

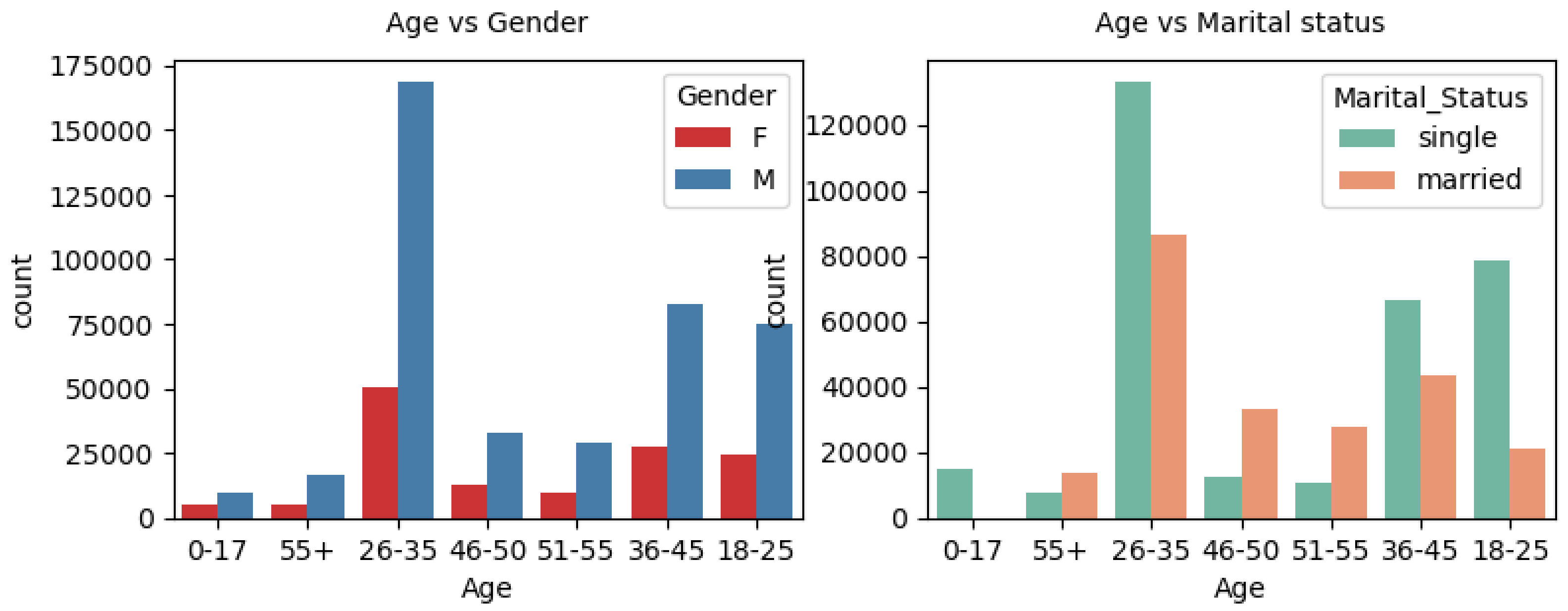


In [21]:
```
#Analysis data of Stay_In_Current_City_Years column
plt.figure(figsize = (6,3))
sns.countplot(data = data, x = data["Stay_In_Current_City_Years"].sort_values(),palette='ch:s=-.25,rot=-.25')
#sns.displot(data = data, x = data["Stay_In_Current_City_Years"],kde= True)
plt.title("Customer distribution over Stay_In_Current_City_Years field", pad=5, fontsize=10)
plt.show()
```

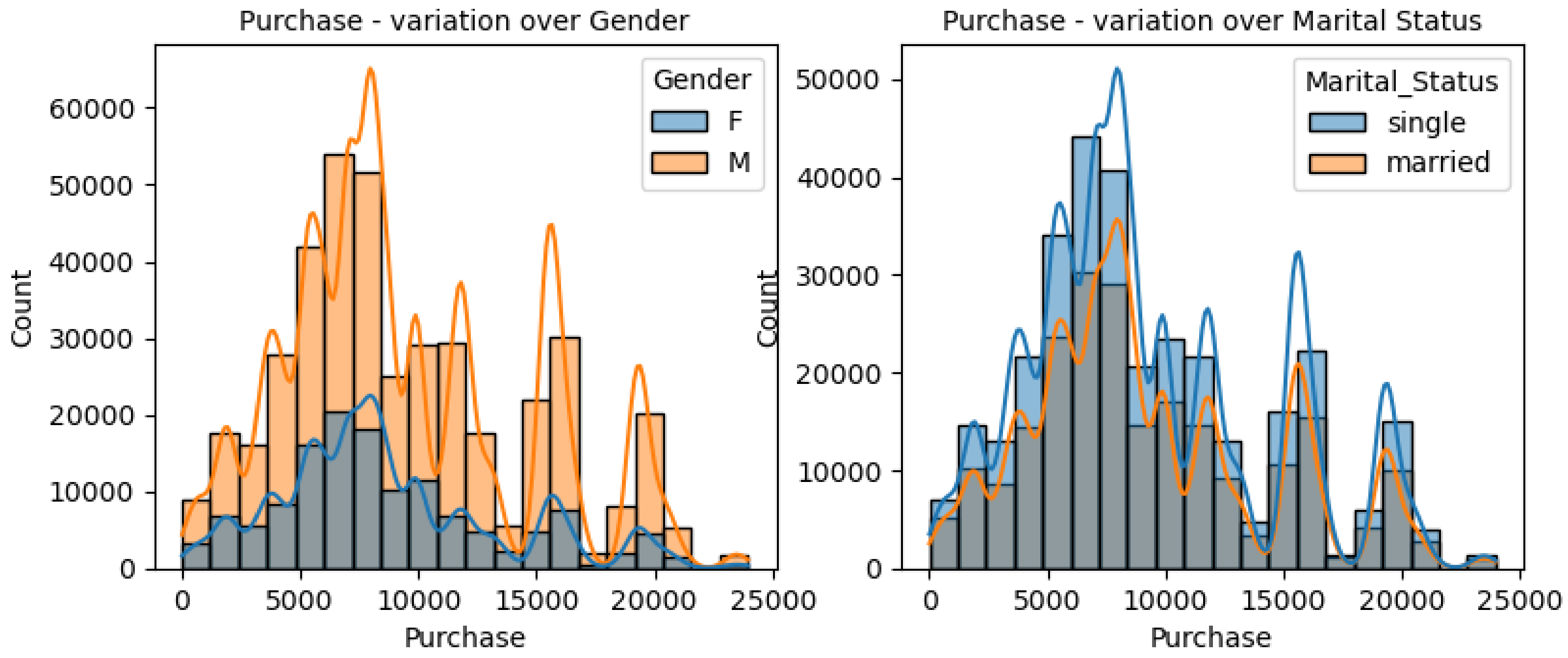


Bivariate Analysis

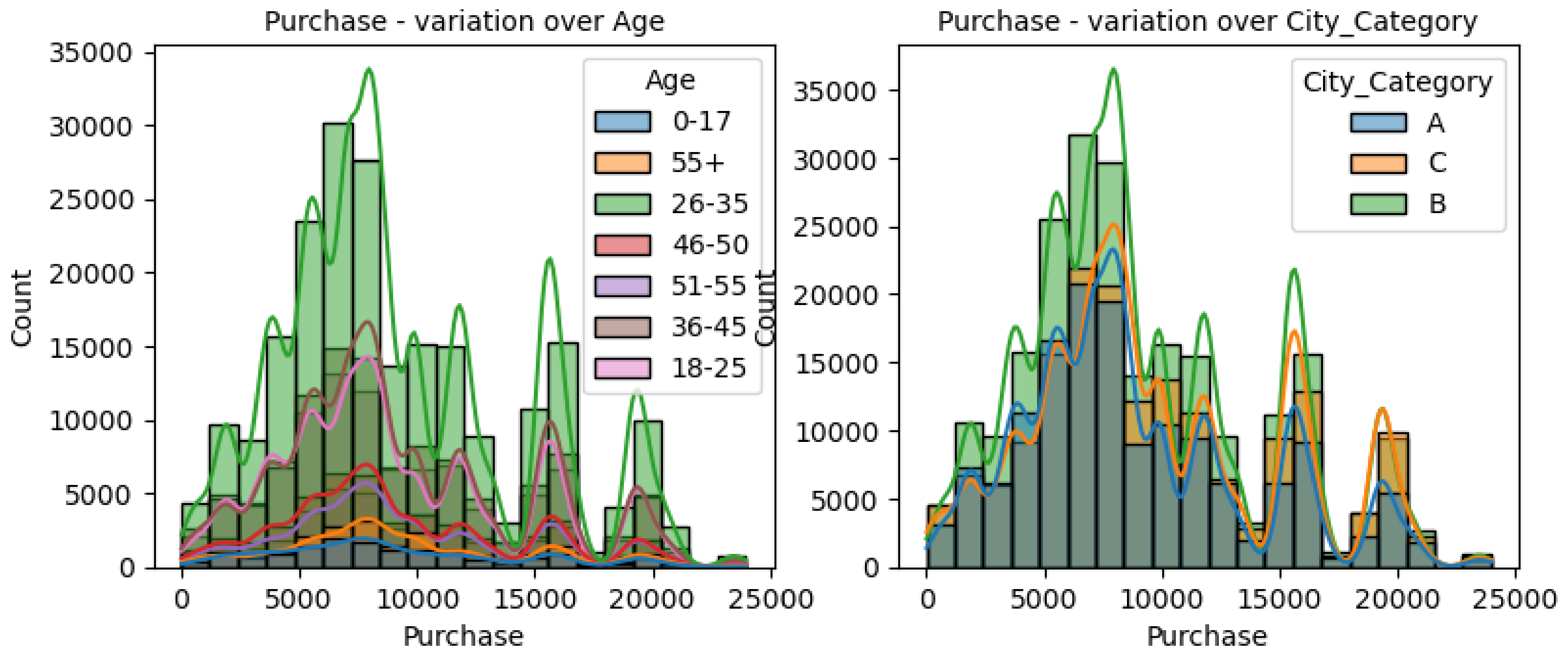
```
In [22]: #Graphs/Plots for distribution of Age over Gender and Marital Status
         plt.figure(figsize = (9,3))
         plt.subplot(1,2,1)
         sns.countplot(data = data, x = "Age", hue = "Gender",palette = "Set1")
         plt.title("Age vs Gender", pad=10, fontsize=10)
         plt.subplot(1,2,2)
         sns.countplot(data = data, x = "Age", hue = "Marital_Status",palette = "Set2")
         plt.title("Age vs Marital status", pad=10, fontsize=10)
         plt.show()

         # Maximum number of customers are from age-group 26-35, in that range, males are higher than
         # females and single customers are higher than married customers.
         # Males are higher than females in all age-brackets.
         # In age-group brackets 36-45 and 18-25 also, number of males is quite higher than females.
         # Similar trend can be seen for single and married customers. Single customers are more
         # than married customers.
```



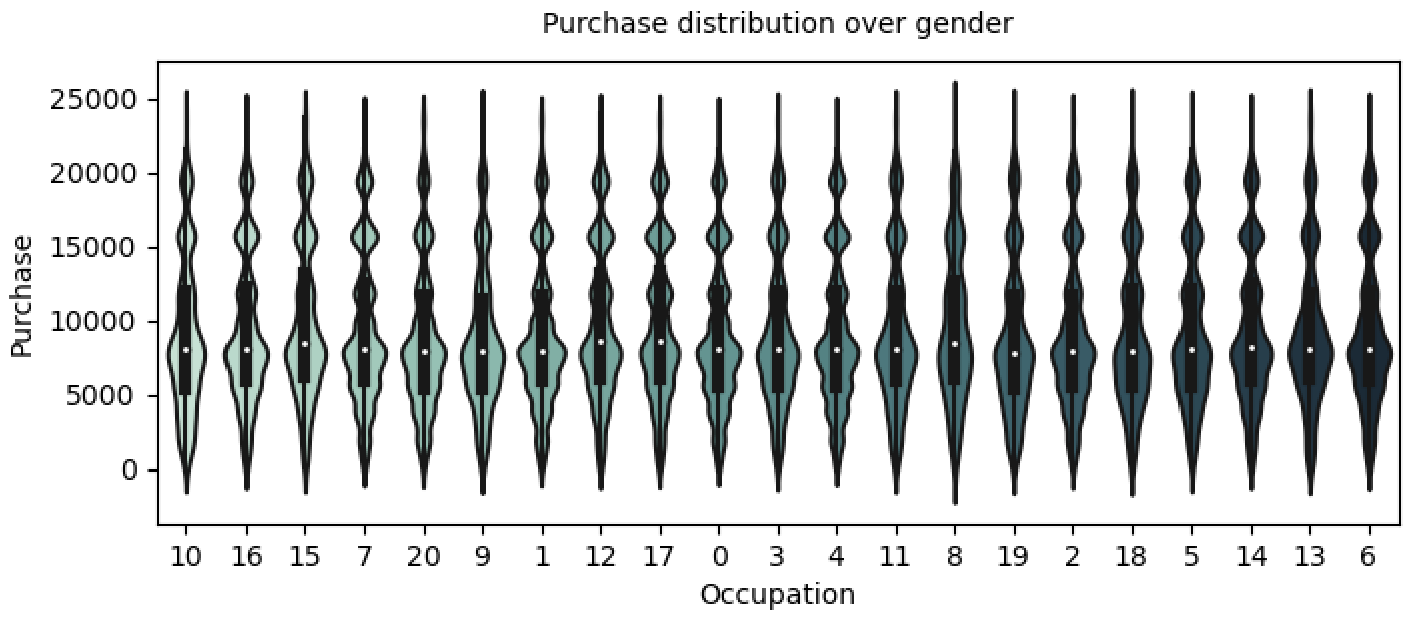
```
In [23]: # Variation of Purchase field over Gender and Marital Status
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
fig.subplots_adjust(top=1)
sns.histplot(x='Purchase',kde='True',data=data, hue='Gender',bins = 20,ax=axis[0])
axis[0].set_title("Purchase - variation over Gender", pad=5, fontsize=10)
sns.histplot(x='Purchase',kde='True',data=data, hue='Marital_Status',bins = 20, ax=axis[1])
axis[1].set_title("Purchase - variation over Marital Status", pad=5, fontsize=10)
plt.show()

# It can be clearly seen from the graph that purchasing capability of males are far higher than
# females.
# Similar trend can be seen for marital status plot also, expenses of single customers are more
# than married customers.
```



```
In [24]: # Variation of Purchase field over Age and city category
         fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
         fig.subplots_adjust(top=1)
         sns.histplot(x='Purchase',kde='True',data=data, hue='Age',bins = 20, ax=axis[0])
         axis[0].set_title("Purchase - variation over Age", pad=5, fontsize=10)
         sns.histplot(x='Purchase',kde='True',data=data, hue='City_Category',bins = 20,ax=axis[1])
         axis[1].set_title("Purchase - variation over City_Category", pad=5, fontsize=10)
         plt.show()

         # Customers of age-group 26-35 are more likely to spend in comparison with others.
         # customers from city B are spending highest among all three cities, followed by customers
         # of city A.
```

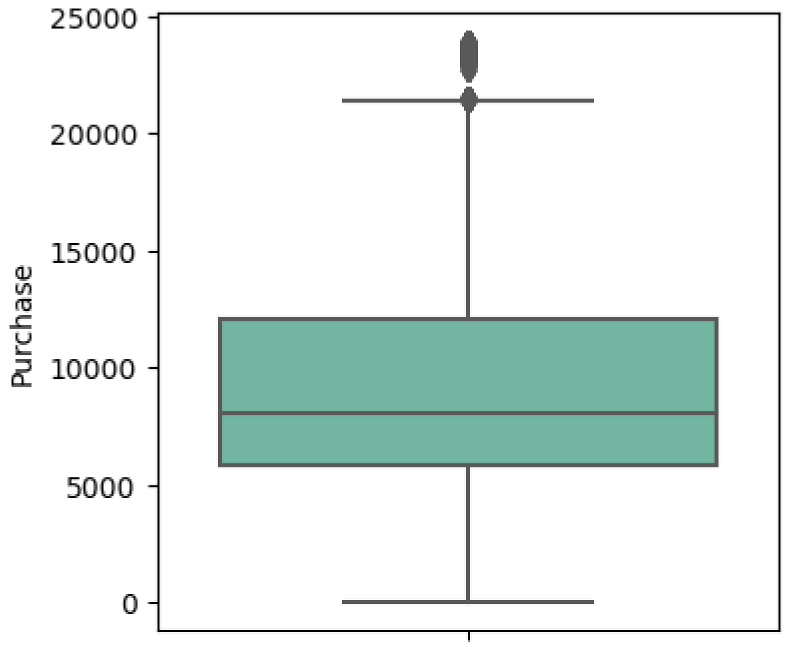


```
In [25]: plt.figure(figsize = (8,3))
         sns.violinplot(data = data, x='Occupation', y = 'Purchase',palette='ch:s=-.2,rot=-.25')
         plt.title("Purchase distribution over gender", pad=10, fontsize=10)
         plt.show()
```



Outliers

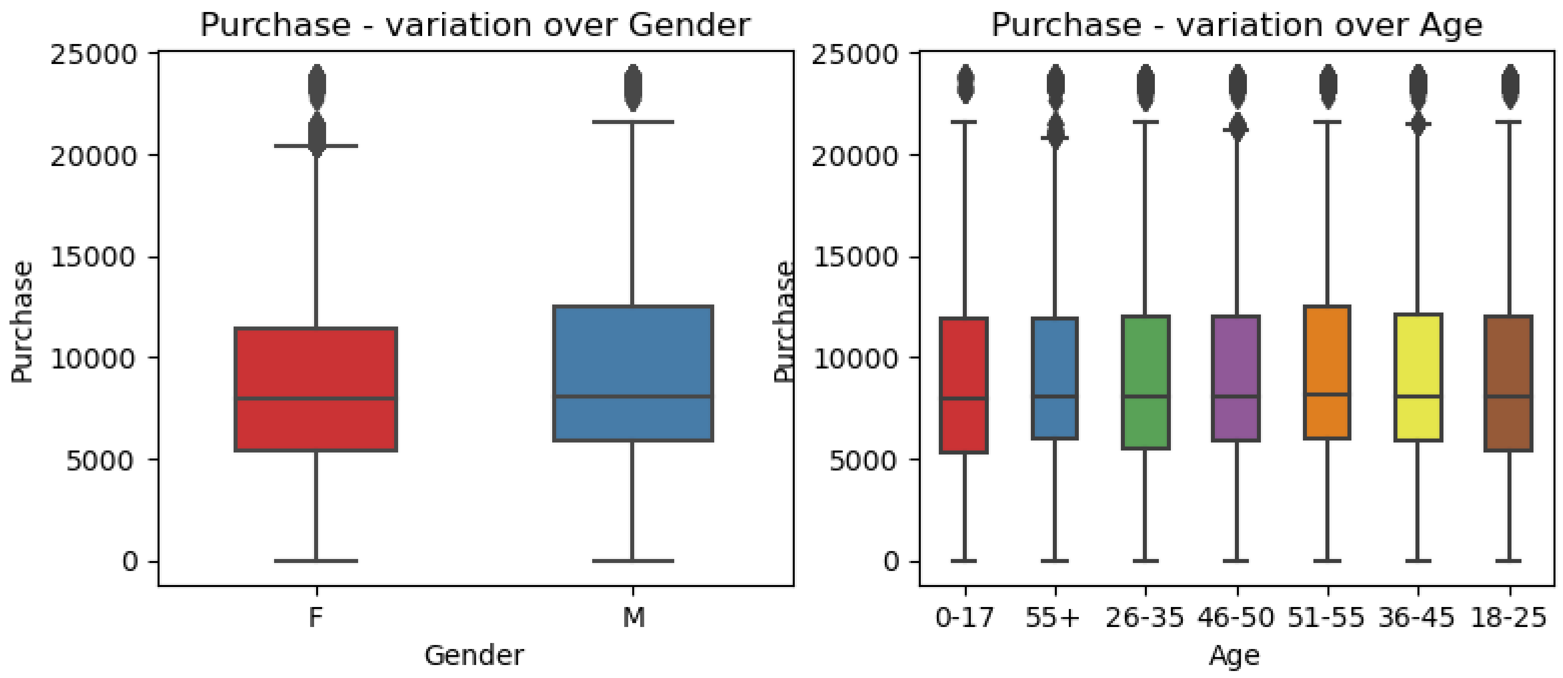
```
In [27]: #Outlier in Purchase Field
fig, axis = plt.subplots(nrows=1, ncols=1, figsize=(4, 4))
sns.boxplot(y = data['Purchase'], palette='Set2')
plt.show()
```



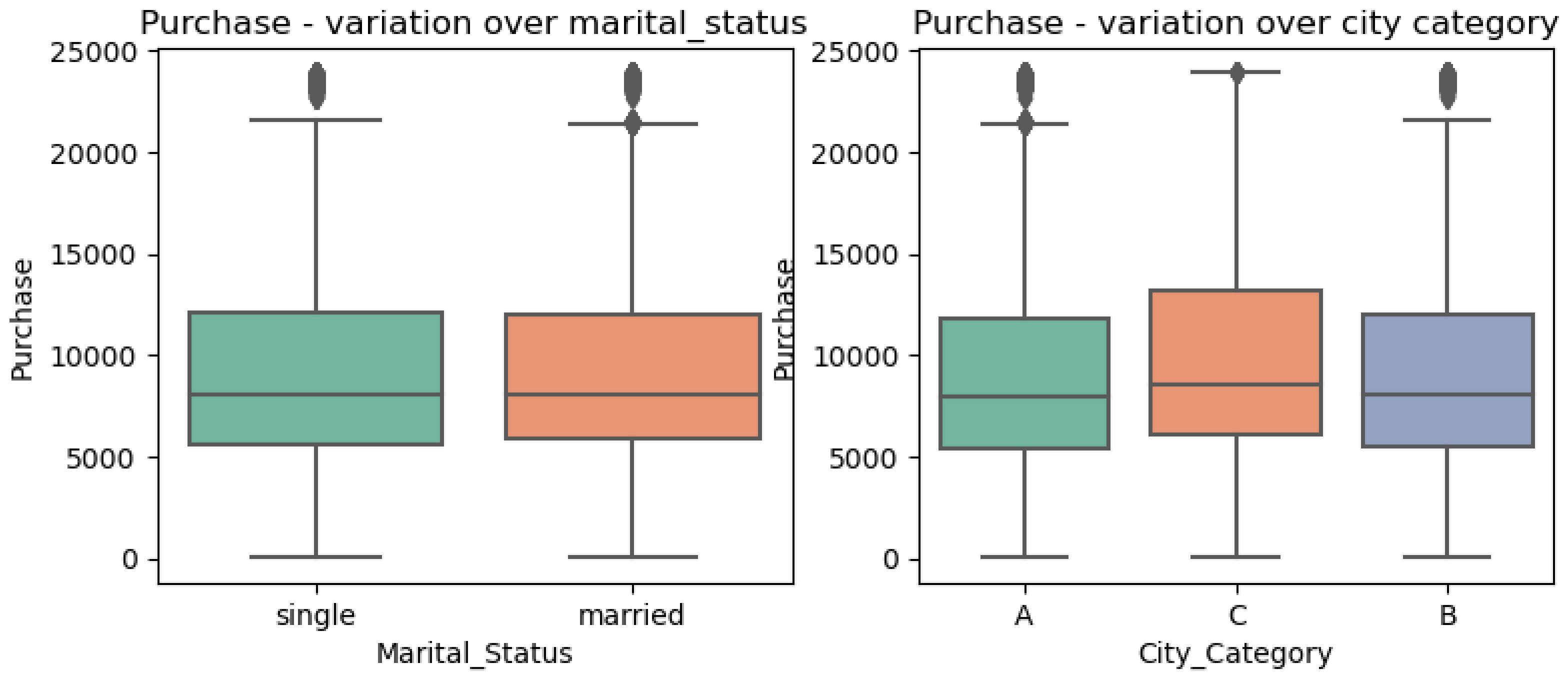
```
1. It can be observe that there are lot of outliers in purchase field.
2. Before treating outliers, it is important to note that on treating outliers, it should not affect
   other fields also.
   Hence, it is necessary to find outliers of purchase field with respect to other fields also.
```

Outliers of purchase field with respect to other fields

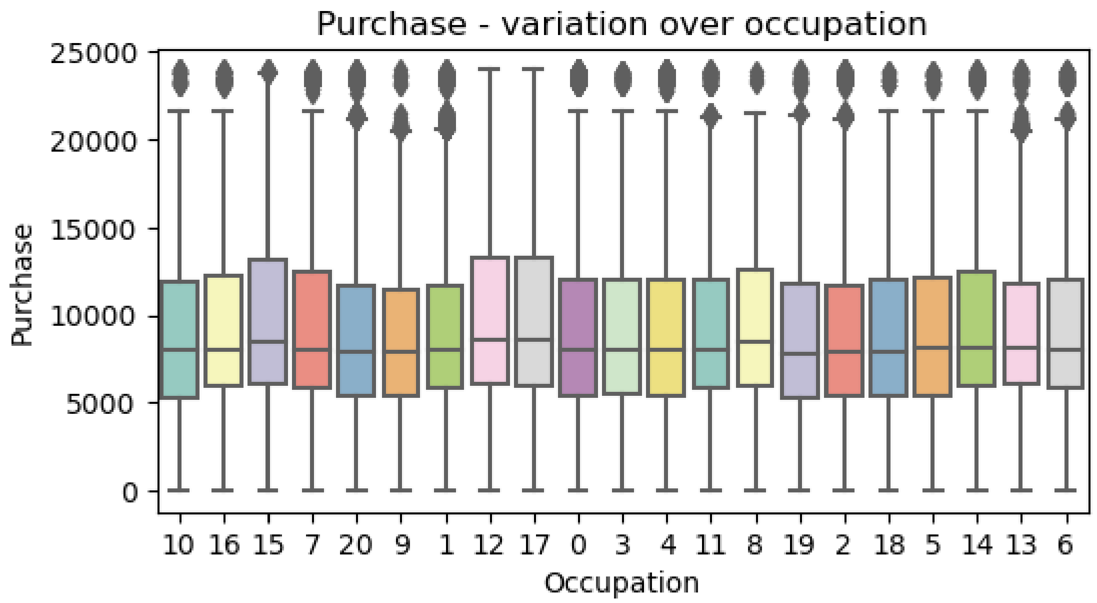
In [28]:
```python
#Outliers in Purchase field with respect to gender and age
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
fig.subplots_adjust(top=1)

sns.boxplot(x=data["Gender"], y = data['Purchase'], palette='Set1',width=0.5,ax=axis[0])
axis[0].set_title("Purchase - variation over Gender", pad=5, fontsize=12)

sns.boxplot(x=data["Age"], y = data['Purchase'], palette='Set1',width=0.5, ax=axis[1])
axis[1].set_title("Purchase - variation over Age", pad=5, fontsize=12)
plt.show()
```



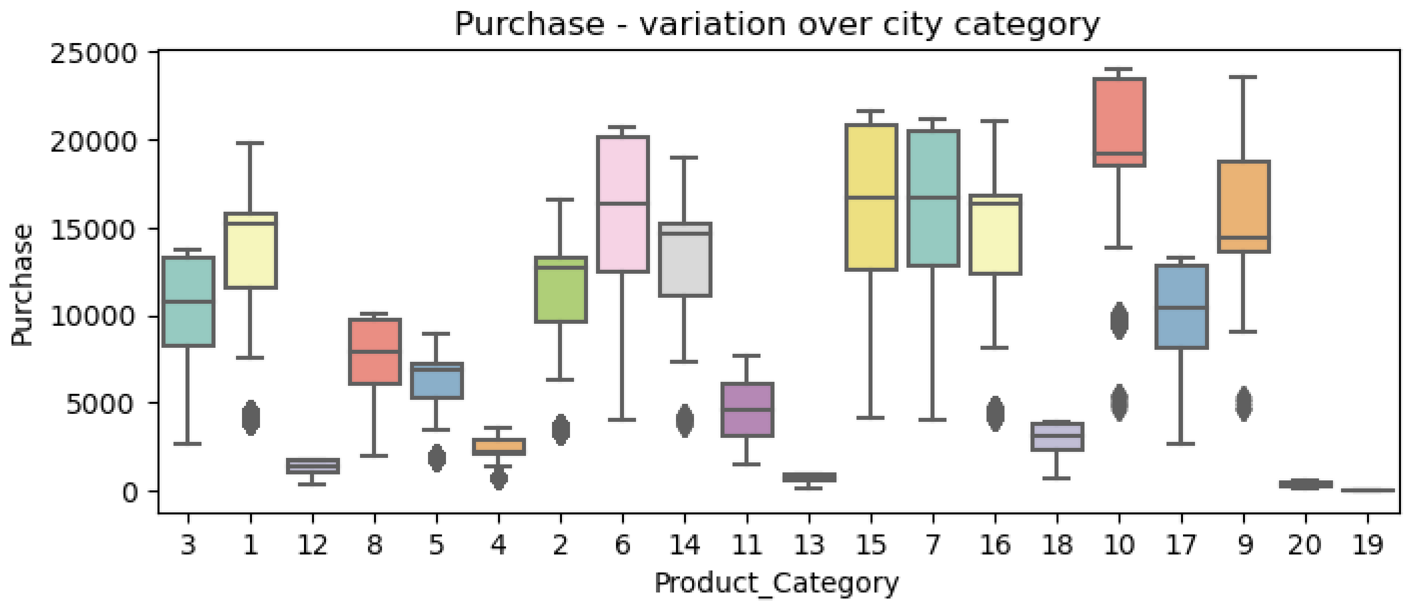
```
In [29]: fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(9, 3))
         fig.subplots_adjust(top=1)

         sns.boxplot(x=data["Marital_Status"], y = data['Purchase'], palette='Set2',ax=axis[0])
         axis[0].set_title("Purchase - variation over marital_status", pad=5, fontsize=12)

         sns.boxplot(x=data["City_Category"], y = data['Purchase'], palette='Set2', ax=axis[1])
         axis[1].set_title("Purchase - variation over city category", pad=5, fontsize=12)
         plt.show()
```



```
In [30]: fig, axis = plt.subplots(nrows=1, ncols=1, figsize=(6, 3))
         sns.boxplot(x=data["Occupation"], y = data['Purchase'], palette='Set3')
         plt.title("Purchase - variation over occupation", pad=5, fontsize=12)
         plt.show()
```



```
In [31]: fig, axis = plt.subplots(nrows=1, ncols=1, figsize=(8, 3))
         sns.boxplot(x=data["Product_Category"], y = data['Purchase'], palette='Set3')
         plt.title("Purchase - variation over city category", pad=5, fontsize=12)
         plt.show()
```



It can be observe from above graphs that every field has outlier in terms of purchase field.
Hence, now outliers can be deleted or replace with mean or median values.

Treating outliers

```
In [32]: # Calculating IQR, upper and lower bound for Income in dataset
         Q1 = np.percentile(data["Purchase"], 25,interpolation = 'midpoint')
         Q3 = np.percentile(data["Purchase"], 75,interpolation = 'midpoint')
         IQR = Q3 - Q1

         upper=Q3+1.5*IQR
         lower=Q1-1.5*IQR

         print("Upper Bound:",upper)
         print("Lower Bound:",lower)

         display(data[data["Purchase"]>=upper][["User_ID","Purchase"]])
         display(data[data["Purchase"]<=lower][["User_ID","Purchase"]])
```

```
Upper Bound: 21400.5
Lower Bound: -3523.5
```

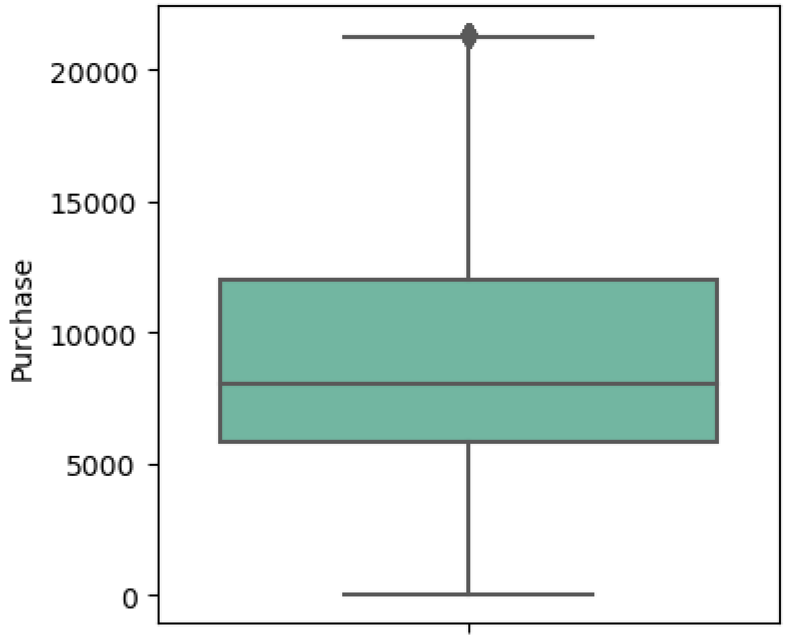
	User_ID	Purchase
343	1000058	23603
375	1000062	23792
652	1000126	23233
736	1000139	23595
1041	1000175	23341
...	...	...
544488	1005815	23753
544704	1005847	23724
544743	1005852	23529
545663	1006002	23663
545787	1006018	23496

2677 rows × 2 columns

User_ID	Purchase

```
In [33]: outlier_data = data[data["Purchase"]>=upper].index

         for i in outlier_data:
             data["Purchase"].iloc[outlier_data] = data["Purchase"].median()

         # Outlier values have been replaced with median values.
```

```
In [34]: #Checking again outliers
         fig, axis = plt.subplots(nrows=1, ncols=1, figsize=(4, 4))
         sns.boxplot(y = data['Purchase'], palette='Set2')
         plt.show()

         # It can be seen now, there is no more outlier outside upper whisker.
```



Distribution of Purchase field with respect to other fields

Distinct users

In [35]:
```
unique_users = pd.DataFrame(data["User_ID"].value_counts()).reset_index()
unique_users.columns = ["Distinct user","Count"]
unique_users = unique_users.sort_values(by = "Distinct user")
unique_users

# There are total 5891 distinct users in the dataset.
```

Out[35]:

	Distinct user	Count
3782	1000001	35
2209	1000002	77
4137	1000003	29
5557	1000004	14
1626	1000005	106
...	...	...
55	1006036	514
1387	1006037	122
5749	1006038	12
2299	1006039	74
835	1006040	180

5891 rows × 2 columns

Distinct Products

In [36]:
```
unique_products = pd.DataFrame(data["Product_ID"].value_counts()).reset_index()
unique_products.columns = ["Distinct Product","Count"]
unique_products = unique_products.sort_values(by = "Distinct Product")
unique_products

# There are total 3631 unique products in the dataset that have been sold.
```

Out[36]:

	Distinct Product	Count
29	P00000142	1152
414	P00000242	376
711	P00000342	244
1585	P00000442	92
1145	P00000542	149
...	...	...
879	P0099442	200
2952	P0099642	13
1290	P0099742	126
1494	P0099842	102
2893	P0099942	14

3631 rows × 2 columns

creating new columns for gender and marital status field with integer/numeric datatype

```
In [37]: data["Gender_num"] = data['Gender'].replace(["M", "F"], [-1, 1])
         data["Marital_num"] = data["Marital_Status"].replace(["single", "married"], [-1, 1])
         data.groupby(['User_ID']).agg({"Gender_num":"count","Marital_num":"count"}).head(2)
```

```
Out[37]:
```

	Gender_num	Marital_num
User_ID		
1000001	35	35
1000002	77	77

Number of distinct users over gender, marital_status and Age field

```
In [38]: data_distinct_user1 = data.groupby(["User_ID"])[["Gender","Age","Occupation","City_Category",
                                                   "Stay_In_Current_City_Years","Marital_Status",
                                                   "Product_Category","Gender_num","Marital_num",
                                                    "Purchase"]]
         data_distinct_user = pd.DataFrame(data_distinct_user1.first()).reset_index()
         distinct_user_grouped_gender = data_distinct_user.groupby(["Gender"]).agg(
             {"Gender_num":"count"}).reset_index()
         distinct_user_grouped_marital_status = data_distinct_user.groupby(
                                                  ["Marital_Status"]).agg({"Marital_num":"count"}).reset_index()
         distinct_user_grouped_age = data_distinct_user.groupby(["Age"]).agg({"Age":"count"})
         display(distinct_user_grouped_gender)
         display(distinct_user_grouped_marital_status)
         display(distinct_user_grouped_age)
```

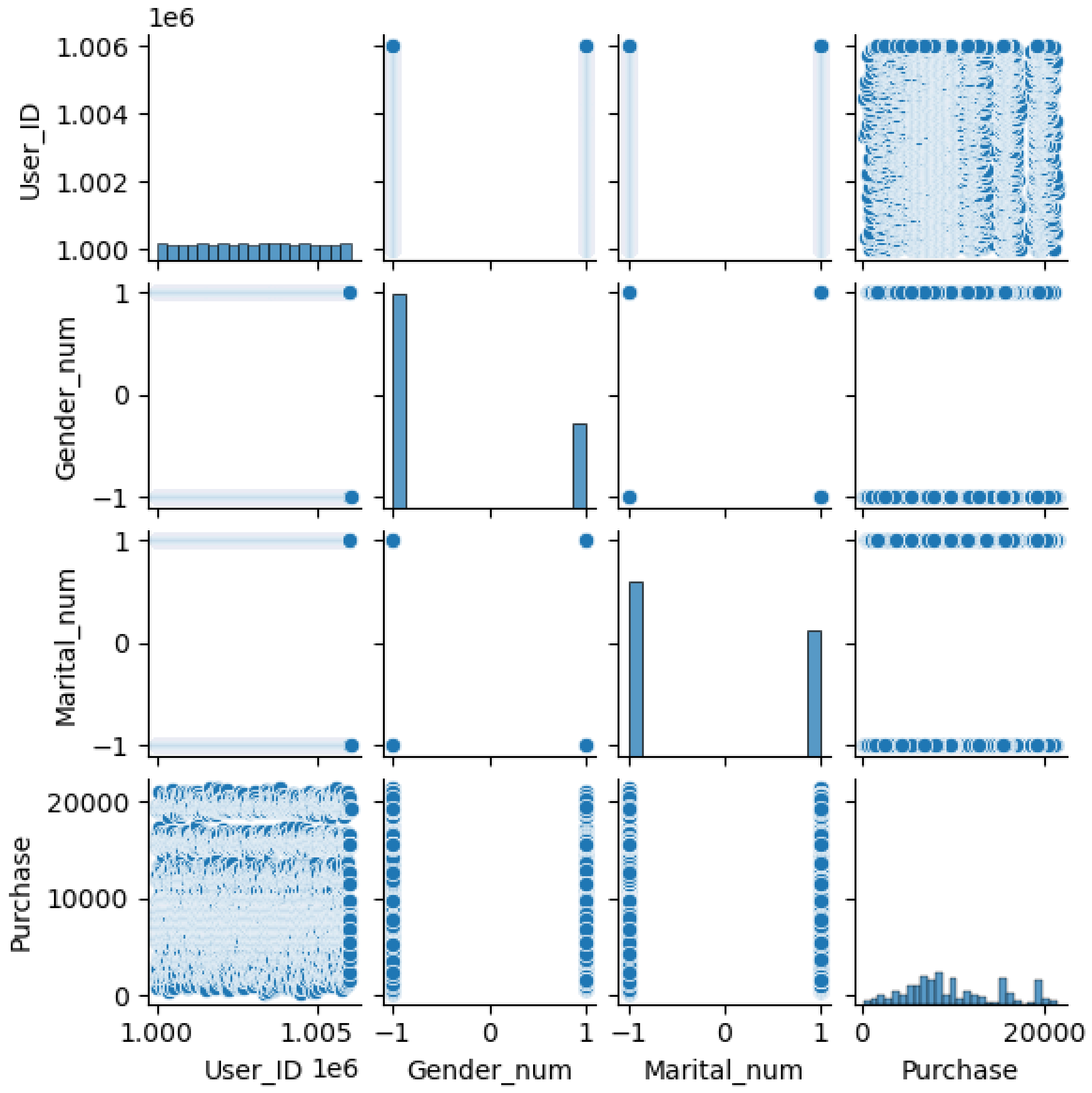
	Gender	Gender_num
0	F	1666
1	M	4225

	Marital_Status	Marital_num
0	married	2474
1	single	3417

	Age
Age	
0-17	218
18-25	1069
26-35	2053
36-45	1167
46-50	531
51-55	481
55+	372

```
1. There are total 1666 unique female customers and 4225 unique male customers in dataset.
2. There are total 2474 unique customers who are married while 3417 unique customers are single.
3. Maximum 2053 unique customers are there, who belong to age-group 26-35 while minimum
   218 unique customers belong to age-group 0-17.
```

In [95]:
```python
# Relation of all fields for distinct users
sns.pairplot(data_distinct_user,palette='ch:s=-.2,rot=-.25',height=1.5, aspect=1)
plt.show()
```



Purchase amount distribution over Gender, Marital Status and Age

In [40]:
```
purchase_gender = data.groupby(["Gender"]).agg({ "Purchase":"sum"}).reset_index()
purchase_marital_status = data.groupby(["Marital_Status"]).agg({ "Purchase":"sum"}).reset_index()
purchase_age = data.groupby(["Age"]).agg({ "Purchase":"sum"}).reset_index()


df1 = pd.crosstab(index= purchase_gender["Gender"], columns= purchase_gender["Purchase"].sum,
                  margins=True, normalize='index')
df2 = pd.crosstab(index= purchase_marital_status["Marital_Status"],
                  columns= purchase_marital_status["Purchase"], margins=True, normalize='index')

display(purchase_gender)
display(purchase_marital_status)
display(purchase_age)
```

	Gender	Purchase
0	F	1177238934
1	M	3877906451

	Marital_Status	Purchase
0	married	2068825682
1	single	2986319703

	Age	Purchase
0	0-17	133858754
1	18-25	908949890
2	26-35	2017704849
3	36-45	1017353398
4	46-50	416923513
5	51-55	362373011
6	55+	197981970

```
1. Male customers are spending more than female customers.
2. Single customers tend to make purchase more than married customers.
3. Customers with age-range 26 to 35 are making more purchases, followed by age group of 36-45,
   in comparison with other age-groups.
4. Customers with age-group 0-17 are buying less products and making less purchase
5. It can be observe that purchasing capability of two age groups (26-35 and 36-45) together is
```

higher than others.

Marginal Probabilities for Gender, Age, City category and Marital Status

In [41]:
```python
data_male = data[data["Gender"] == "M"]
data_female = data[data["Gender"] == "F"]
data_single = data[data["Marital_Status"] == "single"]
data_married = data[data["Marital_Status"] == "married"]
data_city_A = data[data["City_Category"] == "A"]
data_city_B = data[data["City_Category"] == "B"]
data_city_C = data[data["City_Category"] == "C"]

print("P(male):",round(len(data_male)/len(data),2))
print("P(female):",round(len(data_female)/len(data),2))
print("P(single):",round(len(data_single)/len(data),2))
print("P(married):",round(len(data_married)/len(data),2))
print("P(city_A):",round(len(data_city_A)/len(data),2))
print("P(city_B):",round(len(data_city_B)/len(data),2))
print("P(city_C):",round(len(data_city_C)/len(data),2))
```

```
P(male): 0.75
P(female): 0.25
P(single): 0.59
P(married): 0.41
P(city_A): 0.27
P(city_B): 0.42
P(city_C): 0.31
```

```
In [42]: data_age_0_to_17 =data[data["Age"] == "0-17"]
data_age_18_to_25 =data[data["Age"] == "18-25"]
data_age_26_to_35 =data[data["Age"] == "26-35"]
data_age_36_to_45 =data[data["Age"] == "36-45"]
data_age_46_to_50 =data[data["Age"] == "46-50"]
data_age_51_to_55 =data[data["Age"] == "51-55"]
data_age_55 =data[data["Age"] == "55+"]

print("P(age_0_to_17):",round(len(data_age_0_to_17)/len(data),2))
print("P(age_18_to_25):",round(len(data_age_18_to_25)/len(data),2))
print("P(age_26_to_35):",round(len(data_age_26_to_35)/len(data),2))
print("P(age_36_to_45):",round(len(data_age_36_to_45)/len(data),2))
print("P(age_46_to_50):",round(len(data_age_46_to_50)/len(data),2))
print("P(age_51_to_55):",round(len(data_age_51_to_55)/len(data),2))
print("P(age_55):",round(len(data_age_55)/len(data),2))
```

```
P(age_0_to_17): 0.03
P(age_18_to_25): 0.18
P(age_26_to_35): 0.4
P(age_36_to_45): 0.2
P(age_46_to_50): 0.08
P(age_51_to_55): 0.07
P(age_55): 0.04
```

Conditional Probabilities

Given customers belong to age group (26-35 and 36-45), probability of customers being male or female

Given customers belong to age group (26-35 and 36-45), probability of customers being married or single

In [108]:
```
data_age_group = data[(data["Age"] == "26-35") | (data["Age"] == "36-45")]
data_age_group_male = data_age_group[data_age_group["Gender"] == "M"]
data_age_group_female = data_age_group[data_age_group["Gender"] == "F"]
print("P(Male|age_26_to_45):",round(len(data_age_group_male)/len(data_age_group),2))
print("P(Female|age_26_to_45):",round(len(data_age_group_female)/len(data_age_group),2))
data_age_group_married = data_age_group[data_age_group["Marital_Status"] == "married"]
data_age_group_single = data_age_group[data_age_group["Marital_Status"] == "single"]
print("P(married|age_26_to_45):",round(len(data_age_group_married)/len(data_age_group),2))
print("P(single|age_26_to_45):",round(len(data_age_group_single)/len(data_age_group),2))
```

```
P(Male|age_26_to_45): 0.76
P(Female|age_26_to_45): 0.24
P(married|age_26_to_45): 0.39
P(single|age_26_to_45): 0.61
```

```
1. It can be observed that males from age-group 26 to 45 are tend to shop more than females of
   this age group.
2. Single customers from age group 26 to 45 are tend to shop more than married customers of this
   age group.
```

Given gender of customer, probability of being married or single

In [107]:
```
data_male_married = data_male[data_male["Marital_Status"] == "married"]
data_male_single = data_male[data_male["Marital_Status"] == "single"]
print("P(married|male):",round(len(data_male_married)/len(data_male),2))
print("P(single|male):",round(len(data_male_single)/len(data_male),2))
data_female_married = data_female[data_female["Marital_Status"] == "married"]
data_female_single = data_female[data_female["Marital_Status"] == "single"]
print("P(married|female):",round(len(data_female_married)/len(data_female),2))
print("P(single|female):",round(len(data_female_single)/len(data_female),2))
```

```
P(married|male): 0.41
P(single|male): 0.59
P(married|female): 0.42
P(single|female): 0.58
```

```
    It can be seen that no matter whether customer is male or female, single customers are more
```

```
likely to increase the purchase.
```

Given marital status of customers, probability being male or female

In [109]:
```
data_married_male = data_married[data_married["Gender"] == "M"]
data_married_female = data_married[data_married["Gender"] == "F"]
print("P(male|married):",round(len(data_married_male)/len(data_married),2))
print("P(female|married):",round(len(data_married_female)/len(data_married),2))
data_single_male = data_single[data_single["Gender"] == "M"]
data_single_female = data_single[data_single["Gender"] == "F"]
print("P(male|single):",round(len(data_single_male)/len(data_single),2))
print("P(female|single):",round(len(data_single_female)/len(data_single),2))
```

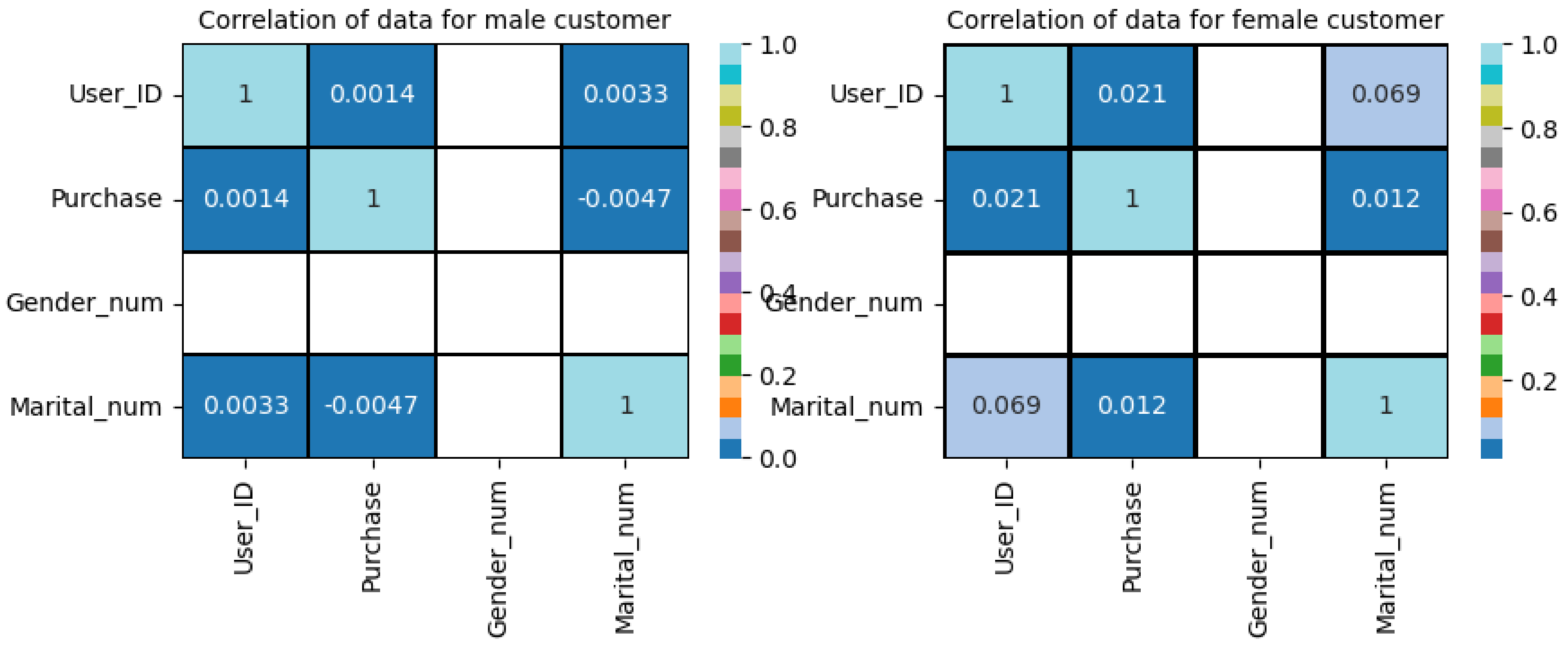
```
P(male|married): 0.75
P(female|married): 0.25
P(male|single): 0.76
P(female|single): 0.24
```

```
It can be seen that no matter whether customer is single or married, males are likely to make
more purchase than females.
```

Multivariate Analysis

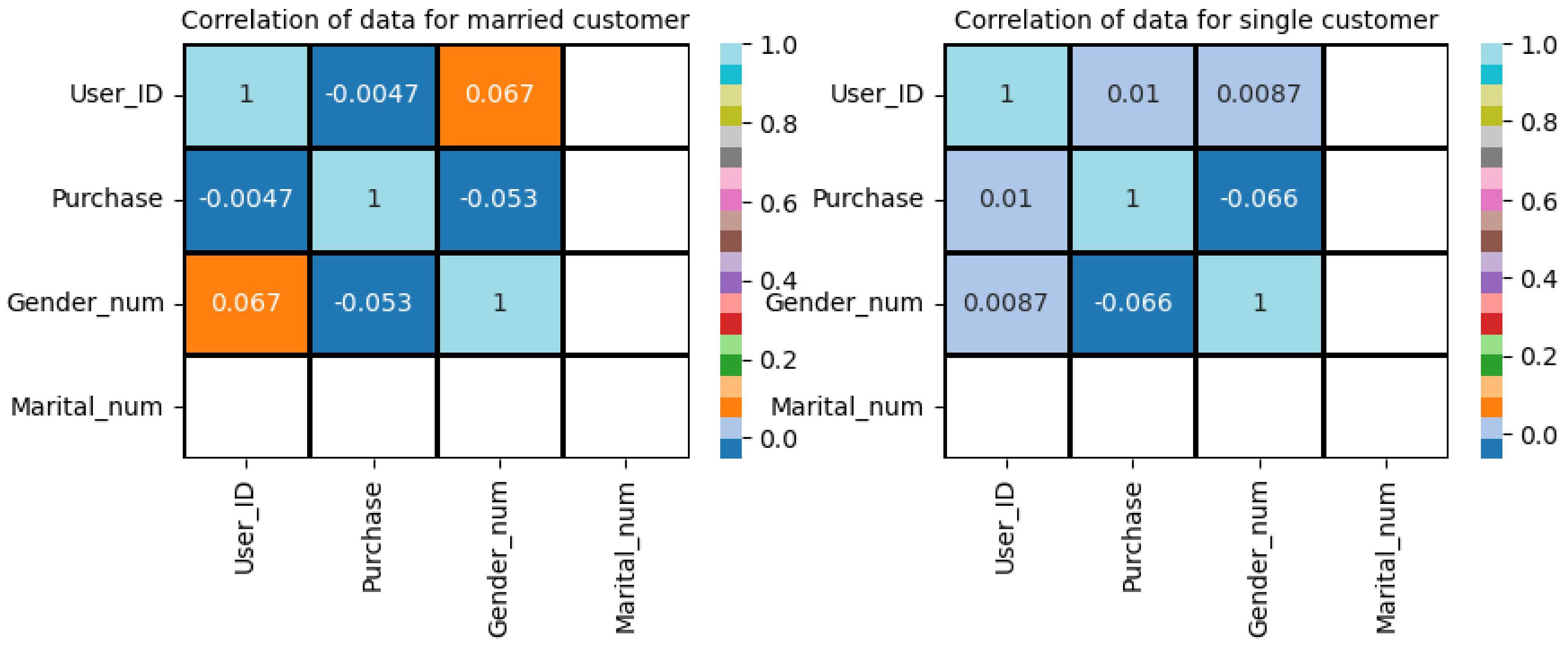
Given customer is male or female, relation of all fields

```
In [110]: plt.figure(figsize = (10,3))
          plt.subplot(1,2,1)
          sns.heatmap(data_male.corr(),linewidths=0.01, linecolor= 'black', annot = True,
                      cbar=True, cmap = "tab20")
          plt.title("Correlation of data for male customer", fontsize = 10)
          plt.subplot(1,2,2)
          sns.heatmap(data_female.corr(),linewidths=1, linecolor= 'black', annot = True,
                      cbar=True, cmap = "tab20")
          plt.title("Correlation of data for female customer", fontsize = 10)
          plt.show()
```



Given marital status of customers, relation of all fields

```
In [111]: plt.figure(figsize = (10,3))
          plt.subplot(1,2,1)
          sns.heatmap(data_married.corr(),linewidths=1, linecolor= 'black', annot = True,
                      cbar=True, cmap = "tab20")
          plt.title("Correlation of data for married customer", fontsize = 10)
          plt.subplot(1,2,2)
          sns.heatmap(data_single.corr(),linewidths=1, linecolor= 'black', annot = True,
                      cbar=True, cmap = "tab20")
          plt.title("Correlation of data for single customer", fontsize = 10)
          plt.show()
```



Statistical Analysis

Calculation of Statistical Parameters for Gender field

In [43]:
```
data_gender = data.groupby('Gender')['Purchase']

def percentile(n):
    def percentile_(x):
        return np.percentile(x, n)
    percentile_.__name__ = 'percentile_%s' % n
    return percentile_

data_gender_stat = data_gender.agg([np.mean, np.std, np.median,np.var, np.min, np.max,
            percentile(25),percentile(50), percentile(75)]).reset_index()
data_gender_stat
```

Out[43]:

	Gender	mean	std	median	var	amin	amax	percentile_25	percentile_50	percentile_75
0	F	8668.342555	4669.080996	7914.0	2.180032e+07	12	21398	5433.0	7914.0	11035.0
1	M	9361.067475	4997.469307	8078.0	2.497470e+07	12	21399	5863.0	8078.0	12183.0

```
1. The mean purchase for female customers is around 8668 with standard deviation of 4669 while
   mean purchase of male customers is 9361 with standard deviation of 4997.
2. The maximum purchase by female and male are 21398 and 21399 respectively.
3. The minimum purchase by female and male are 12 and 12 respectively.
4. 75% of females have made purchases below 11035.
5. 75% of males have made purchases below 12183.
```

Calculation of Statistical Parameters for Marital Status field

In [44]:
```
data_marital_status = data.groupby('Marital_Status')['Purchase']

def percentile(n):
    def percentile_(x):
        return np.percentile(x, n)
    percentile_.__name__ = 'percentile_%s' % n
    return percentile_

data_marital_status_stat = data_marital_status.agg([np.mean, np.std, np.median,np.var, np.min, np.max,
            percentile(25),percentile(50), percentile(75)]).reset_index()
data_marital_status_stat
```

Out[44]:

	Marital_Status	mean	std	median	var	amin	amax	percentile_25	percentile_50	percentile_75
0	married	9181.029667	4912.898655	8047.0	2.413657e+07	12	21398	5843.0	8047.0	11993.0
1	single	9196.287706	4937.585642	8044.0	2.437975e+07	12	21399	5605.0	8044.0	12017.0

```
1. The mean purchase for married customers is around 9181 with standard deviation of 4912 while
   mean purchase of single customers is 9196 with standard deviation of 4937.
2. The maximum purchase by married and single customers are 21398 and 21399 respectively.
3. The minimum purchase by married and single customers are 12 and 12 respectively.
4. 75% of married customers have made purchases below 11993.
5. 75% of single customers have made purchases below 12017.
```

Calculation of Statistical Parameters for Age field

```
In [45]: data_age = data.groupby('Age')['Purchase']

         def percentile(n):
             def percentile_(x):
                 return np.percentile(x, n)
             percentile_.__name__ = 'percentile_%s' % n
             return percentile_

         data_age_stat = data_age.agg([np.mean, np.std, np.median,np.var, np.min, np.max,
                     percentile(25),percentile(50), percentile(75)]).reset_index()
         data_age_stat
```

Out[45]:

	Age	mean	std	median	var	amin	amax	percentile_25	percentile_50	percentile_75
0	0-17	8863.644153	5018.690448	7986.0	2.518725e+07	12	21342	5328.0	7986.0	11817.75
1	18-25	9120.508629	4971.061556	8027.0	2.471145e+07	12	21398	5415.0	8027.0	11997.00
2	26-35	9188.635252	4927.549399	8030.0	2.428074e+07	12	21398	5475.0	8030.0	12008.00
3	36-45	9247.574359	4915.007949	8047.0	2.415730e+07	12	21399	5876.0	8047.0	12053.00
4	46-50	9122.853176	4854.281141	8036.0	2.356405e+07	12	21391	5888.0	8036.0	11942.00
5	51-55	9412.041531	4935.193726	8104.0	2.435614e+07	12	21388	6017.0	8104.0	12104.00
6	55+	9206.750837	4842.194327	8080.0	2.344685e+07	12	21345	6018.0	8080.0	11806.00

1. The mean purchase is highest for customers of age-group 51-55 with standard deviation of 4935, followed by customers of age-group 36-45 with standard deviation of 4915.
2. The minimum mean purchase, that is, 8863, has been recorded for customers with age-group 0-17 with standard deviation of 5018.
   The standard deviation is maximum for this age-group.
4. 75% of customers with age group from 26 to 45 and from 51 to 55 have made high purchases.
5. Although, the customers of age-group 51-55 are less in count, but they have made high purchases.
   In other words, these customers are equally important.

Calculation of different Confidence Intervals for different Fields

90%, 95%, and 99% Confidence Interval for Gender

In [46]:
```
mean_male = data_gender_stat.loc[1]['mean']
mean_female = data_gender_stat.loc[0]['mean']
std_male = data_gender_stat.loc[1]['std']
std_female = data_gender_stat.loc[0]['std']
n = 5000000
std_error_male =round(std_male/(n**(1/2)),2)
std_error_female =round(std_female/(n**(1/2)),2)

# z multiplier values for 90%, 95% and 99% confidence interval
z =[1.645,1.96,2.576]
print("standard error for male is:",std_error_male)
print("standard error for female is:",std_error_female)
```

```
standard error for male is: 2.23
standard error for female is: 2.09
```

```
    standard error for males is quite higher than standard error for females.
```

In [47]:
```
interval_90_male = [round(mean_male-(z[0]*std_error_male),2),
                    round(mean_male+(z[0]*std_error_male),2)]
interval_95_male = [round(mean_male-(z[1]*std_error_male),2),
                    round(mean_male+(z[1]*std_error_male),2)]
interval_99_male = [round(mean_male-(z[2]*std_error_male),2),
                    round(mean_male+(z[2]*std_error_male),2)]

interval_90_female = [round(mean_female-(z[0]*std_error_female),2),
                      round(mean_female+(z[0]*std_error_female),2)]
interval_95_female = [round(mean_female-(z[1]*std_error_female),2),
                      round(mean_female+(z[1]*std_error_female),2)]
interval_99_female = [round(mean_female-(z[2]*std_error_female),2),
                      round(mean_female+(z[2]*std_error_female),2)]
```

In [48]:
```
CI_gender = pd.DataFrame()
CI_gender["Gender"] = ["Male","Female"]
CI_gender["90%_interval"] = [interval_90_male,interval_90_female]
CI_gender["95%_interval"] = [interval_95_male,interval_95_female]
CI_gender["99%_interval"] = [interval_99_male,interval_99_female]
CI_gender
```

Out[48]:

	Gender	90%_interval	95%_interval	99%_interval
0	Male	[9357.4, 9364.74]	[9356.7, 9365.44]	[9355.32, 9366.81]
1	Female	[8664.9, 8671.78]	[8664.25, 8672.44]	[8662.96, 8673.73]

Population mean for males and females within 90%, 95%, and 99% can be estimated according to above table as:

1. Population mean for males is found to be within 90% of confidence intervals [9357.4,9364.74]
2. Population mean for males is found to be within 95% of confidence intervals [9356.7, 9365.44]
3. Population mean for males is found to be within 99% of confidence intervals [9355.32, 9366.81]
4. Population mean for females is found to be within 90% of confidence intervals [9355.32, 9366.81]
5. Population mean for females is found to be within 95% of confidence intervals [8664.25, 8672.44]
6. Population mean for females is found to be within 99% of confidence intervals [8662.96, 8673.73]

90%, 95%, and 99% Confidence Interval for Marital Status

In [114]:
```
mean_married = data_marital_status_stat.loc[0]['mean']
mean_single = data_marital_status_stat.loc[1]['mean']
std_married = data_marital_status_stat.loc[0]['std']
std_single = data_marital_status_stat.loc[1]['std']

std_error_married =std_married/(n**0.5)
std_error_single =std_single/(n**0.5)

print("standard error for married customers is:",round(std_error_married,2))
print("standard error for single customers is:",round(std_error_single,2))
```

```
standard error for married customers is: 2.2
standard error for single customers is: 2.21
```

```
    Margin of errors for customers being married or single is almost same.
```

In [50]:
```
interval_90_married = [round(mean_married-((z[0]*std_married)/n**0.5),2),
                       round(mean_married+((z[0]*std_married)/n**0.5),2)]
interval_95_married = [round(mean_married-((z[1]*std_married)/n**0.5),2),
                       round(mean_married+((z[1]*std_married)/n**0.5),2)]
interval_99_married = [round(mean_married-((z[2]*std_married)/n**0.5),2),
                       round(mean_married+((z[2]*std_married)/n**0.5),2)]

interval_90_single = [round(mean_single-((z[0]*std_single)/n**0.5),2),
                      round(mean_single+((z[0]*std_single)/n**0.5),2)]
interval_95_single = [round(mean_single-((z[1]*std_single)/n**0.5),2),
                      round(mean_single+((z[1]*std_single)/n**0.5),2)]
interval_99_single = [round(mean_single-((z[2]*std_single)/n**0.5),2),
                      round(mean_single+((z[2]*std_single)/n**0.5),2)]
```

In [51]:
```
CI_marital_status = pd.DataFrame()
CI_marital_status["marital_status"] = ["married","single"]
CI_marital_status["90%_interval"] = [interval_90_married,interval_90_single]
CI_marital_status["95%_interval"] = [interval_95_married,interval_95_single]
CI_marital_status["99%_interval"] = [interval_99_married,interval_99_single]
CI_marital_status
```

Out[51]:

	marital_status	90%_interval	95%_interval	99%_interval
0	married	[9177.42, 9184.64]	[9176.72, 9185.34]	[9175.37, 9186.69]
1	single	[9192.66, 9199.92]	[9191.96, 9200.62]	[9190.6, 9201.98]

```
    Population mean for married and single customers within 90%, 95%, and 99% can be
    estimated according to above table.
```

90%, 95%, and 99% Confidence Interval for Age

In [53]:
```
mean_0_17 = data_age_stat.loc[0]['mean']
mean_18_25 = data_age_stat.loc[1]['mean']
mean_26_35 = data_age_stat.loc[2]['mean']
mean_36_45 = data_age_stat.loc[3]['mean']
mean_46_50 = data_age_stat.loc[4]['mean']
mean_51_55 = data_age_stat.loc[5]['mean']
mean_55 = data_age_stat.loc[6]['mean']

std_0_17 = data_age_stat.loc[0]['std']
std_18_25 = data_age_stat.loc[1]['std']
std_26_35 = data_age_stat.loc[2]['std']
std_36_45 = data_age_stat.loc[3]['std']
std_46_50 = data_age_stat.loc[4]['std']
std_51_55 = data_age_stat.loc[5]['std']
std_55 = data_age_stat.loc[6]['std']

std_error_0_17 =std_0_17/(n**0.5)
std_error_18_25 =std_18_25/(n**0.5)
std_error_26_35 =std_26_35/(n**0.5)
std_error_36_45 =std_36_45/(n**0.5)
std_error_46_50 =std_46_50/(n**0.5)
std_error_51_55 =std_51_55/(n**0.5)
std_error_55 =std_55/(n**0.5)
```

In [120]:
```python
print("standard error for age-groups:")
print("age-group 0-17:",round(std_error_0_17,2))
print("age-group 18-25:",round(std_error_18_25,2))
print("age-group 26-35:",round(std_error_26_35,2))
print("age-group 36-45:",round(std_error_36_45,2))
print("age-group 46-50:",round(std_error_46_50,2))
print("age-group 51-55:",round(std_error_51_55,2))
print("age-group greater than 55:",round(std_error_55,2))
```

```
standard error for age-groups:
age-group 0-17: 2.24
age-group 18-25: 2.22
age-group 26-35: 2.2
age-group 36-45: 2.2
age-group 46-50: 2.17
age-group 51-55: 2.21
age-group greater than 55: 2.17
```

```
    Standard errors for all age groups come out to be same.
```

In [54]:
```
interval_90_0_17 = [round(mean_0_17-((z[0]*std_0_17)/n**0.5),2),
                    round(mean_0_17+((z[0]*std_0_17)/n**0.5),2)]
interval_95_0_17 = [round(mean_0_17-((z[1]*std_0_17)/n**0.5),2),
                    round(mean_0_17+((z[1]*std_0_17)/n**0.5),2)]
interval_99_0_17 = [round(mean_0_17-((z[2]*std_0_17)/n**0.5),2),
                    round(mean_0_17+((z[2]*std_0_17)/n**0.5),2)]

interval_90_18_25 = [round(mean_18_25-((z[0]*std_18_25)/n**0.5),2),
                      round(mean_18_25+((z[0]*std_18_25)/n**0.5),2)]
interval_95_18_25 = [round(mean_18_25-((z[1]*std_18_25)/n**0.5),2),
                      round(mean_18_25+((z[1]*std_18_25)/n**0.5),2)]
interval_99_18_25 = [round(mean_18_25-((z[2]*std_18_25)/n**0.5),2),
                      round(mean_18_25+((z[2]*std_18_25)/n**0.5),2)]

interval_90_26_35 = [round(mean_26_35-((z[0]*std_26_35)/n**0.5),2),
                      round(mean_26_35+((z[0]*std_26_35)/n**0.5),2)]
interval_95_26_35 = [round(mean_26_35-((z[1]*std_26_35)/n**0.5),2),
                      round(mean_26_35+((z[1]*std_26_35)/n**0.5),2)]
interval_99_26_35 = [round(mean_26_35-((z[2]*std_26_35)/n**0.5),2),
                      round(mean_26_35+((z[2]*std_26_35)/n**0.5),2)]

interval_90_36_45 = [round(mean_36_45-((z[0]*std_36_45)/n**0.5),2),
                      round(mean_36_45+((z[0]*std_36_45)/n**0.5),2)]
interval_95_36_45 = [round(mean_36_45-((z[1]*std_36_45)/n**0.5),2),
                      round(mean_36_45+((z[1]*std_36_45)/n**0.5),2)]
interval_99_36_45 = [round(mean_36_45-((z[2]*std_36_45)/n**0.5),2),
                      round(mean_36_45+((z[2]*std_36_45)/n**0.5),2)]
```

```
In [55]: interval_90_46_50 = [round(mean_46_50-((z[0]*std_46_50)/n**0.5),2),
                              round(mean_46_50+((z[0]*std_46_50)/n**0.5),2)]
         interval_95_46_50 = [round(mean_46_50-((z[1]*std_46_50)/n**0.5),2),
                              round(mean_46_50+((z[1]*std_46_50)/n**0.5),2)]
         interval_99_46_50 = [round(mean_46_50-((z[2]*std_46_50)/n**0.5),2),
                              round(mean_46_50+((z[2]*std_46_50)/n**0.5),2)]

         interval_90_51_55 = [round(mean_51_55-((z[0]*std_51_55)/n**0.5),2),
                              round(mean_51_55+((z[0]*std_51_55)/n**0.5),2)]
         interval_95_51_55 = [round(mean_51_55-((z[1]*std_51_55)/n**0.5),2),
                              round(mean_51_55+((z[1]*std_51_55)/n**0.5),2)]
         interval_99_51_55 = [round(mean_51_55-((z[2]*std_51_55)/n**0.5),2),
                              round(mean_51_55+((z[2]*std_51_55)/n**0.5),2)]

         interval_90_55 = [round(mean_55-((z[0]*std_55)/n**0.5),2),
                              round(mean_55+((z[0]*std_55)/n**0.5),2)]
         interval_95_55 = [round(mean_55-((z[1]*std_55)/n**0.5),2),
                              round(mean_55+((z[1]*std_55)/n**0.5),2)]
         interval_99_55 = [round(mean_55-((z[2]*std_55)/n**0.5),2),
                              round(mean_55+((z[2]*std_55)/n**0.5),2)]
```

In [56]:
```
CI_age = pd.DataFrame()
CI_age["age"] = ["0-17","18-25","26-35","36-45","46-50","51-55",">55"]
CI_age["90%_interval"] = [interval_90_0_17,interval_90_18_25,interval_90_26_35,interval_90_36_45,
                          interval_90_46_50,interval_90_51_55,interval_90_55]
CI_age["95%_interval"] = [interval_95_0_17,interval_95_18_25,interval_95_26_35,interval_95_36_45,
                          interval_95_46_50,interval_95_51_55,interval_90_55]
CI_age["99%_interval"] = [interval_99_0_17,interval_99_18_25,interval_99_26_35,interval_99_36_45,
                          interval_99_46_50,interval_99_51_55,interval_90_55]
CI_age
```

Out[56]:

	age	90%_interval	95%_interval	99%_interval
0	0-17	[8859.95, 8867.34]	[8859.25, 8868.04]	[8857.86, 8869.43]
1	18-25	[9116.85, 9124.17]	[9116.15, 9124.87]	[9114.78, 9126.24]
2	26-35	[9185.01, 9192.26]	[9184.32, 9192.95]	[9182.96, 9194.31]
3	36-45	[9243.96, 9251.19]	[9243.27, 9251.88]	[9241.91, 9253.24]
4	46-50	[9119.28, 9126.42]	[9118.6, 9127.11]	[9117.26, 9128.45]
5	51-55	[9408.41, 9415.67]	[9407.72, 9416.37]	[9406.36, 9417.73]
6	>55	[9203.19, 9210.31]	[9203.19, 9210.31]	[9203.19, 9210.31]

```
Population mean for customers of all age-groups and within 90%, 95%, and 99% can be
estimated according to above table.
```

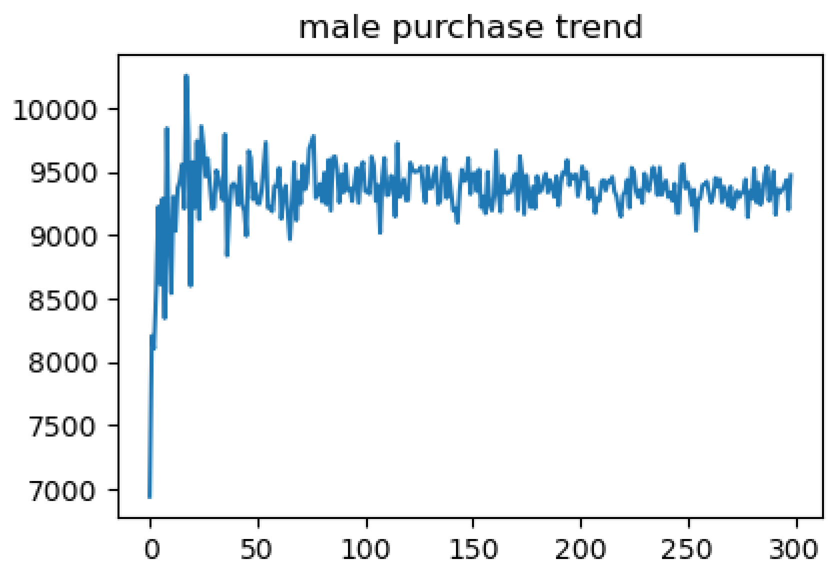
Bootstrapping

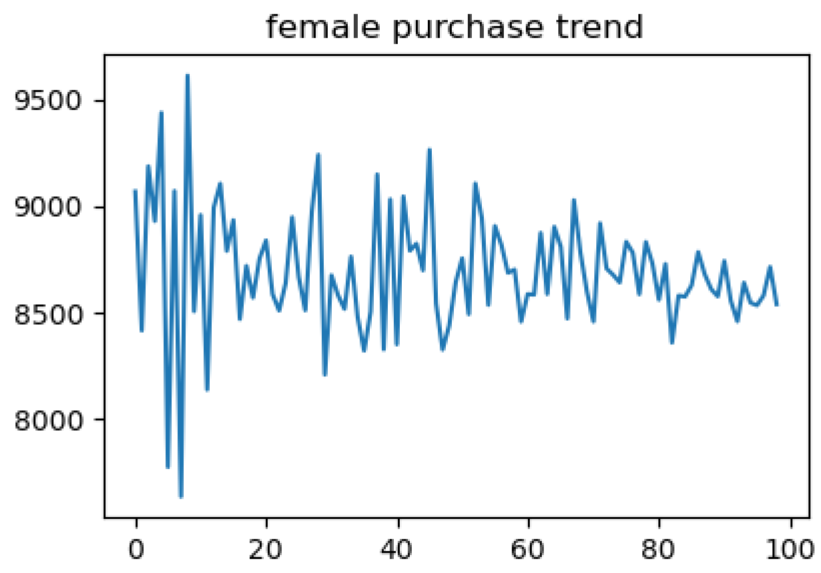
```
In [127]: male_purchase = data_male["Purchase"]
          female_purchase = data_female["Purchase"]

          bootstrapped_samples_male = np.random.choice(male_purchase,size = 1000)
          bootstrapped_samples_female = np.random.choice(female_purchase,size = 1000)

          print("Original purchase mean of males:" ,round(np.mean(male_purchase),3))
          print("Original purchase mean of females:", round(np.mean(female_purchase),3))

          print("Purchase mean of 1000 males:", round(np.mean(bootstrapped_samples_male),3))
          print("Purchase mean of 1000 million females:", round(np.mean(bootstrapped_samples_female),3))
```

```
Original purchase mean of males: 9361.067
Original purchase mean of females: 8668.343
Purchase mean of 1000 males: 9467.918
Purchase mean of 1000 million females: 8684.195
```

Trend for male and female purchases

In [124]:
```
male_purchase_trend = []
for num in range(10,3000,10):
    sample_m = data_male["Purchase"].sample(num)
    sample_mean_m = np.mean(sample_m)
    male_purchase_trend.append(sample_mean_m)

female_purchase_trend = []
for num in range(10,1000,10):
    sample_f = data_female["Purchase"].sample(num)
    sample_mean_f = np.mean(sample_f)
    female_purchase_trend.append(sample_mean_f)

plt.figure(figsize = (10,3))
plt.subplot(1,2,1)
plt.plot(male_purchase_trend)
plt.title("male purchase trend")
plt.subplot(1,2,2)
plt.plot(female_purchase_trend)
plt.title("female purchase trend")
plt.show()

# Notice the plot for female purchase trend is not very central as number of samples taken is less as
# compared to number of samples taken for male purchase trend.
```

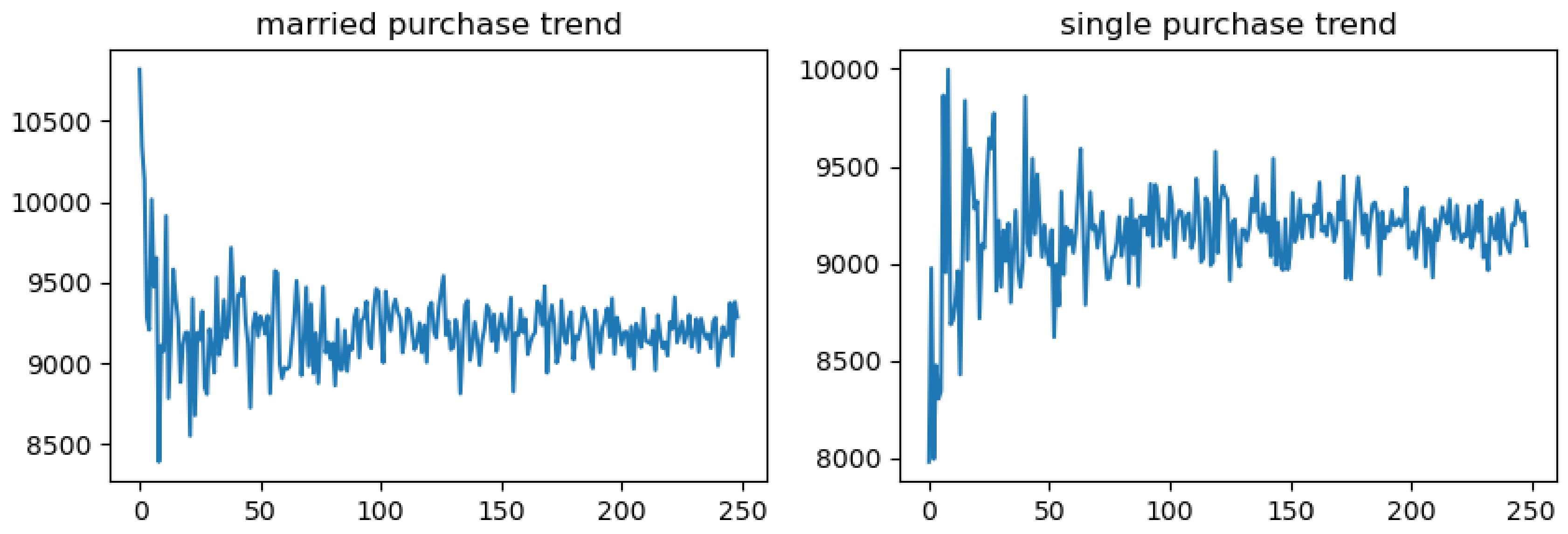




```
The above plot will come out to be different every time the code runs, but it will center around
value around 9400. The center of dispersion will remain same. However, the spread of dispersion
will become lesser as number of samples increases.
Similar trend can be seen for female purchase trend.
```

Trend for married and single customers

In [125]:
```
married_purchase_trend = []
for num in range(10,2500,10):
    sample_mar = data_married["Purchase"].sample(num)
    sample_mean_mar = np.mean(sample_mar)
    married_purchase_trend.append(sample_mean_mar)

single_purchase_trend = []
for num in range(10,2500,10):
    sample_single = data_single["Purchase"].sample(num)
    sample_mean_single = np.mean(sample_single)
    single_purchase_trend.append(sample_mean_single)

plt.figure(figsize = (10,3))
plt.subplot(1,2,1)
plt.plot(married_purchase_trend)
plt.title("married purchase trend")
plt.subplot(1,2,2)
plt.plot(single_purchase_trend)
plt.title("single purchase trend")
plt.show()
```

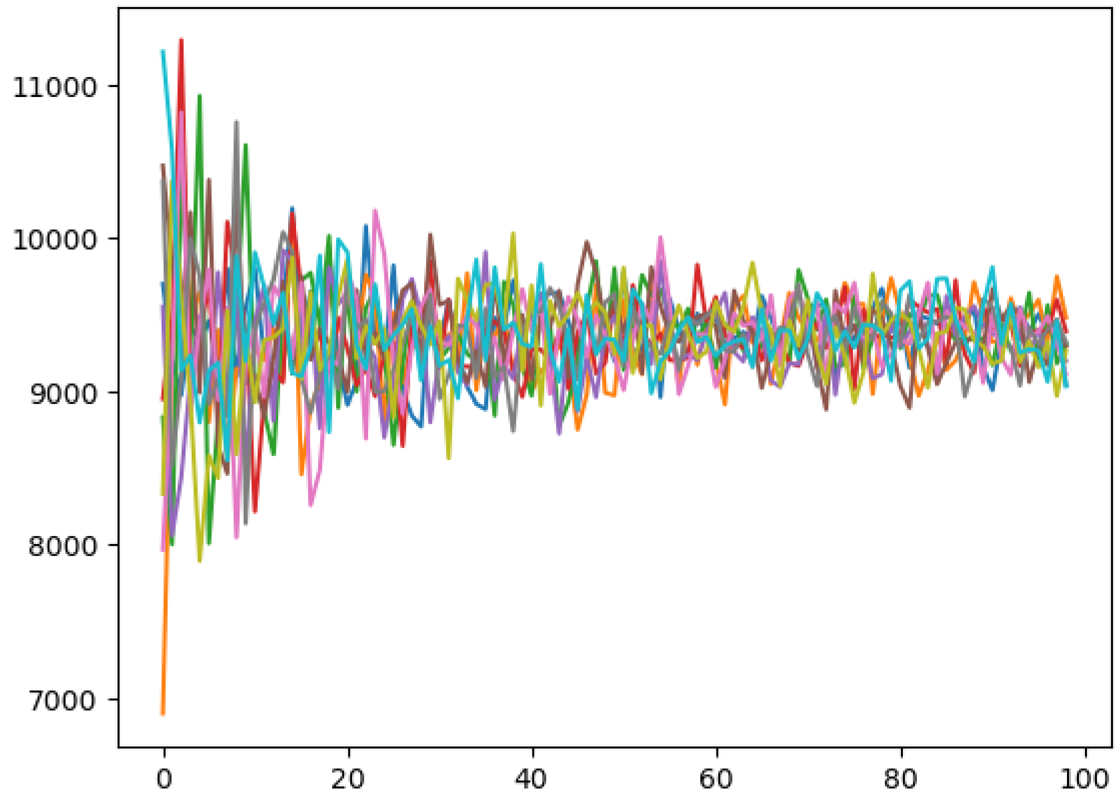


Here also, the plots will be different every time the code runs, but it will center around

```
value of 9400. The center of dispersion will remain same. However, the spread of dispersion
will become lesser as number of samples increases.
Similar trend can be seen for purchase trend of single customers.
```

What if, 10 persons are taking samples ?

In [126]:
```python
male_purchase_trend_sample = []
for person in range(10):
    for num in range(10,1000,10):
        sample_m = data_male["Purchase"].sample(num)
        sample_mean_m = np.mean(sample_m)
        male_purchase_trend_sample.append(sample_mean_m)
    plt.plot(male_purchase_trend_sample)
    male_purchase_trend_sample = []
```



```
For all 10 persons,the variance is high for lesser values, and it becomes less and lesser
as number of samples increases. However, the mean will be centered around the mean value
```

```
    for all the 10 surveys.
```

Insights based on graphs and plots

```
    All the insights based on plot or graph has been attached with corresponding code of the
    plots in commented form.
    Please scroll above to find the same.
```

Insights based on Confidence Intervals

```
1. Standard error for males is quite higher than standard error for females.
2. Margin of errors for customers being married or single is almost same.
3. Margin of errors for customers of all age-groups is almost same.
4. Refer output of code line 48 for confidence Intervals based on gender.
5. Refer output of code line 51 for confidence Intervals based on marital status.
6. Refer output of code line 56 for confidence Intervals based on age groups.
7. The 90% confidence intervals for male and female are quite overlapping.
8. Respective confidence Intervals for married and single customers are not overlapping.
9. The confidence intervals of some age-groups are overlapping with each other.
```

Business Insights

```
1.  Males are likely to spend more than females
2.  Single customers are likely to spend more than married customers.
3.  Customers of age group 26-35 are spending highest, followed by customers of 36-45 age-group.
4.  Maximum customers are from age group 26-35, followed by age-group 36-45 and then 18-25.
    This can be inferred that young and middle-aged adults are more likely to expense in comparison
    with others.
5.  It is interesting to note here that even though percentage of customers belongs to age group
    45 to 55 are less but they are expending more. They tend to buy such products which costs
    high and hence, creating great revenue also.
6.  Customers from city B are spending highest among all three cities, followed by customers of
    city A.
7.  Males from age-group 26 to 45 are tend to shop more than females of this age group.
```

```
8.  Single customers from age group 26 to 45 are tend to shop more than married customers of this
    age group.
9.  No matter whether customer is male or female, single customers are more likely to increase
    the purchase.
10. No matter whether customer is single or married, males are likely to make more purchase
    than females.
```

Answering questions

1. Are women spending more money per transaction than men? Why or Why not?

```
    No, women are not spending more money per transaction than men. As, there are total
    1666 unique female customers and 4225 unique male customers in dataset and expenses made
    by females is 1177238934 and expenses made by males is 3877906451. Therefore, expense ratio
    for females is around 706626 and expense ratio for men is around 917847.
    Expense ratio of men is far greater than of females. Hence, men are spending more money than
    women.
```

2. Confidence intervals and distribution of the mean of the expenses by female and male customers?

```
    Confidence Interval for males:
        90% : [9357.4,9364.74]
        95% : [9356.7, 9365.44]
        99% : [9355.32, 9366.81]
    Confidence Interval for females:
        90% : [9355.32, 9366.81]
        95% : [8664.25, 8672.44]
        99% : [8662.96, 8673.73]
    The mean purchase for female customers is around 8668 with standard deviation of 4669
    while mean purchase of male customers is around 9361 with standard deviation of 4997.
```

3. Are confidence intervals of average male and female spending overlapping? How can Walmart

leverage this conclusion to make changes or improvements?

```
The 90% confidence interval of male spending is overlapping with 90% confidence interval of
female spending. It can be misleading. To avoid such uncertainty, the Walmart can conduct
any Statistical Hypothesis test or closely monitor the statistical mean and the variance
or spread of the dispersion for both males and females.

However, 95% or 99% confidence intervals are not overlapping and one can infer insights
clearly if 95% or 99% confidence interval has been taken in account.
```

4. Results when the same activity is performed for Married vs Unmarried?

```
Single customers are likely to spend more than married customers. The purchasing capacity
of single customer is higher than married customers.

The mean purchase for married customers is around 9181 with standard deviation of 4912 while
mean purchase of single customers is 9196 with standard deviation of 4937.

Confidence Interval for married customers:
    90% : [9177.42, 9184.64]
    95% : [9176.72, 9185.34]
    99% : [9175.37, 9186.69]
Confidence Interval for single customers:
    90% : [9192.66, 9199.92]
    95% : [9191.96, 9200.62]
    99% : [9190.6, 9201.98]

Here, confidence intervals are not overlapping for respective confidence intervals.
```

5. Results when the same activity is performed for Age?

```
Customers of age group 26-35 and 26-45 are highly likely to spend more compare with others.

The mean purchase is highest for customers of age-group 51-55 with standard deviation of 4935,
followed by customers of age-group 36-45 with standard deviation of 4915.
The minimum mean purchase, that is, 8863, has been recorded for customers with age-group 0-17
with standard deviation of 5018. The standard deviation is maximum for this age-group.
```

Refer output of code line 56 for confidence Intervals based on age groups.

The confidence intervals of some age-groups are overlapping with each other. This can be misleading. Walmart needs to conduct any type of statistical hypothesis testing.

Recommendations

1. As males and single customers are more active on Black Friday, Walmart can put more products which can relate to males and single customers in store on this day.
2. Young and middle-aged adults are tend to buy more on Black Friday, Walmart needs to keep those items which are highly related to this age group. For example: Gym equipments, different types of T-shirts, watches and so on.
3. It can be observe that customers of age group 45 to 55 are less in number but they buy expensive items. Therefore, a close observation needed about the products which this group is buying.
4. Walmarts can open their store early in morning and close late in City B as number of customers are more in this city.
5. Occupations masked 0, 4, and 7 can be put on high priority as customers having these occupations are likely to made purchase more.
6. Walmart can prioritize those products which are expected to have good revenue.